How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

Task 16: Implement ChatActivity

Task 17: Implement Send and Receive Messages

Task 18: Implement Chat Pagination

Task 19: Implement send Image Messages

Task 20: Implement search user menu option

Task 21: Implement Chats Fragment

Task 22: Implement Request Fragment

Task 23: Implement Home Screen Widget

Task 24: Implement Land Responsive design

GitHub Username: https://github.com/elbhwashy

ChatApp

Description

Chat with your family and friends.

FAST: ChatApp is the fastest app on the market, connecting people all over the world.

POWERFUL: You can create chats for 10,000 members. It's the perfect tool for hosting communities and coordinating teamwork.

RELIABLE: ChatApp is the most reliable messaging system ever made. It works even on the weakest mobile connections.

SIMPLE: While providing an unprecedented array of features, we are taking great care to keep the interface clean. With its minimalist design, ChatApp is lean and easy to use.

"App is written solely in the Java Programming Language"

1) App keeps all strings in the `strings.xml` file.
2) The app enables RTL layout switching on all layouts.
3) The app includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.

Intended User

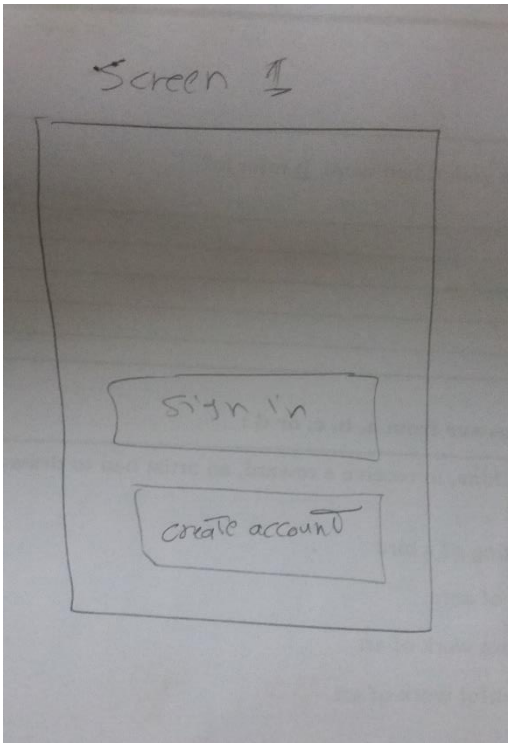
It's the perfect tool for hosting communities and coordinating teamwork. A perfect tool for Family And Friends too.

Features

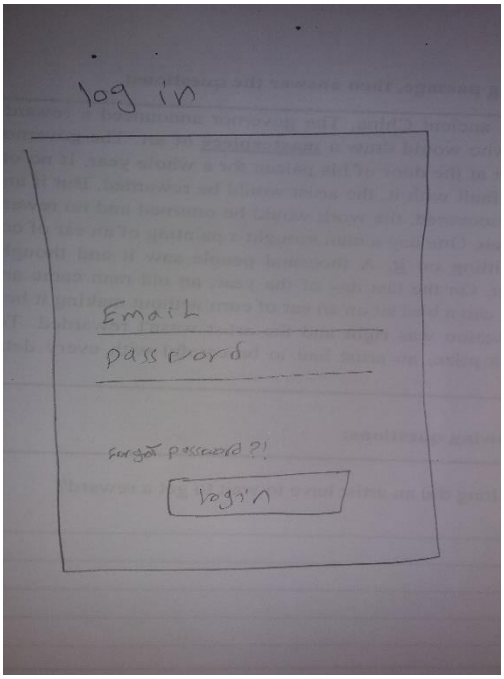
- 1 on 1 chat
- Follow/Unfollow Members
- Invite Friends
- Ask friendship
- Presence System (is people online?)
- Last seen feature
- Users search
- Send images in chat messages
- Notifications

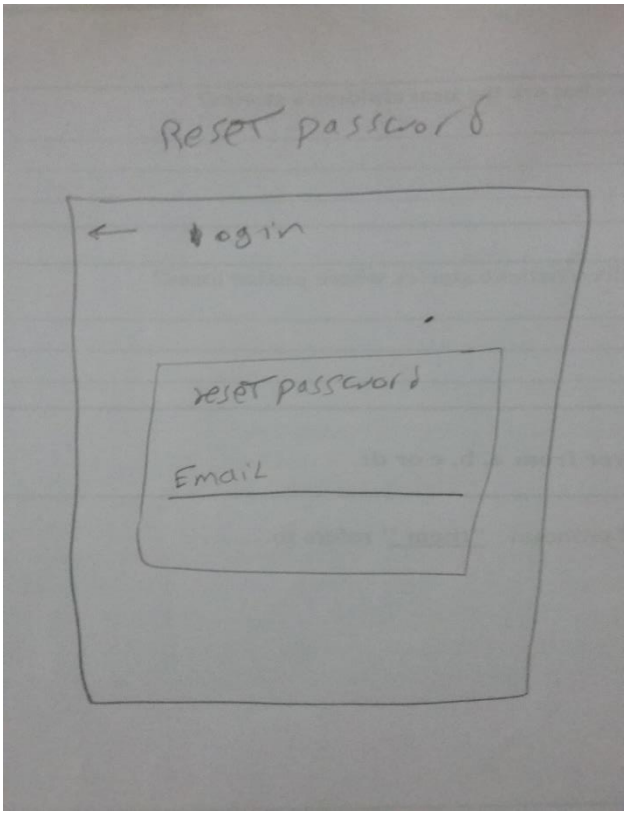
User Interface Mocks

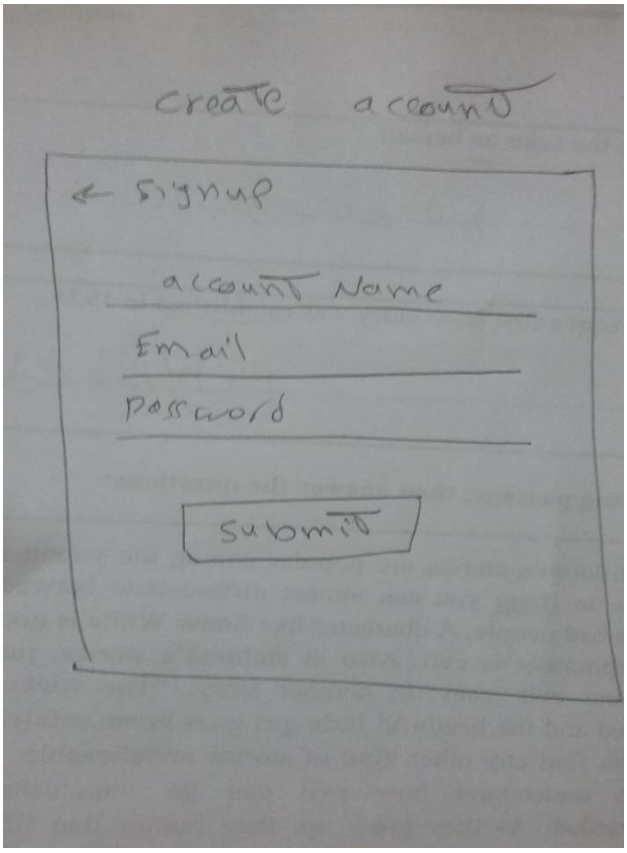
Screen 1

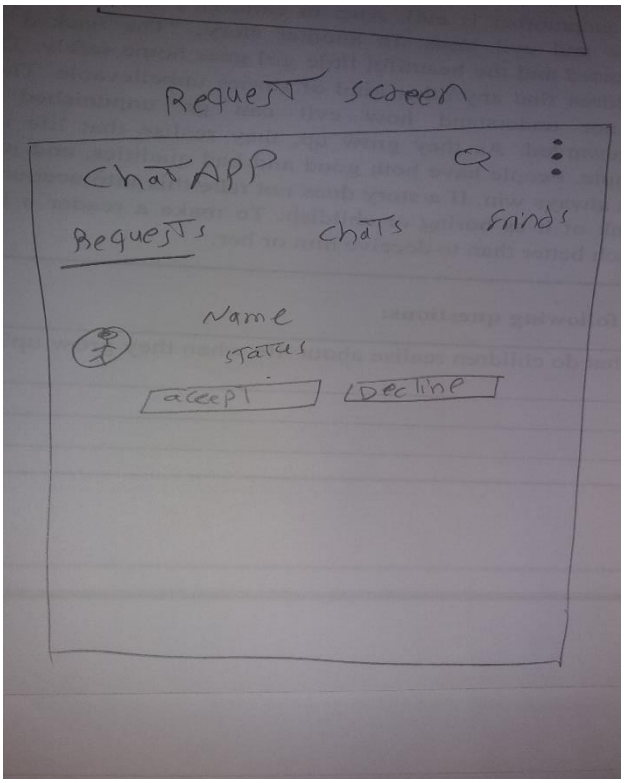


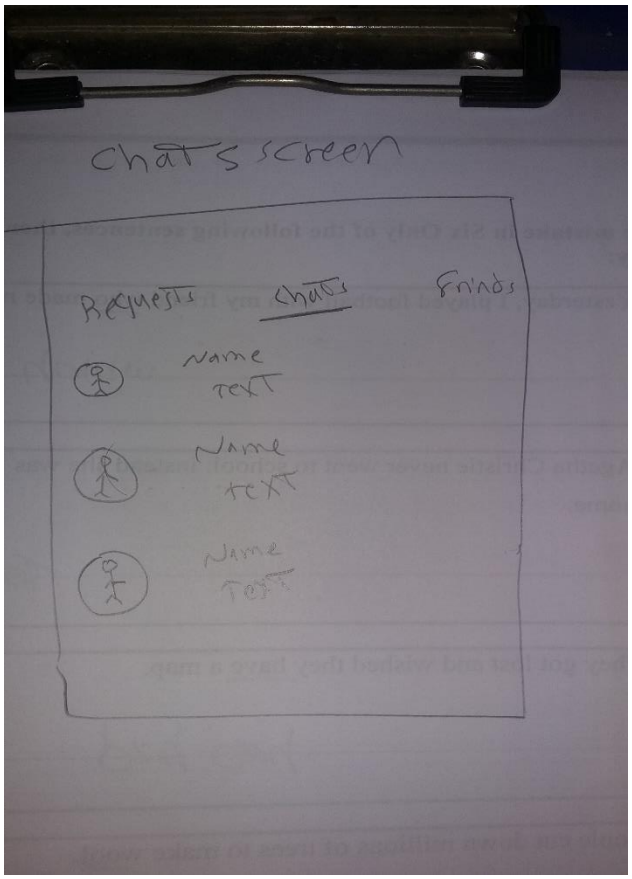
Screen 2

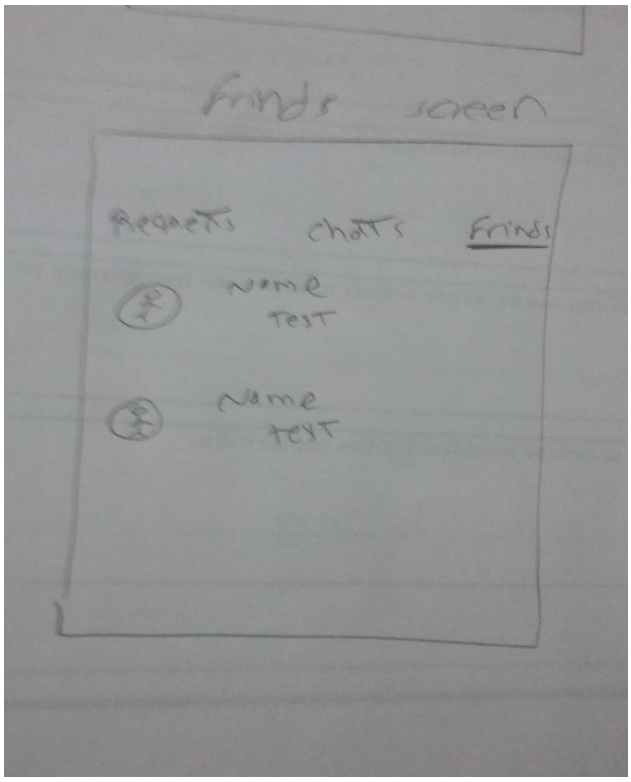


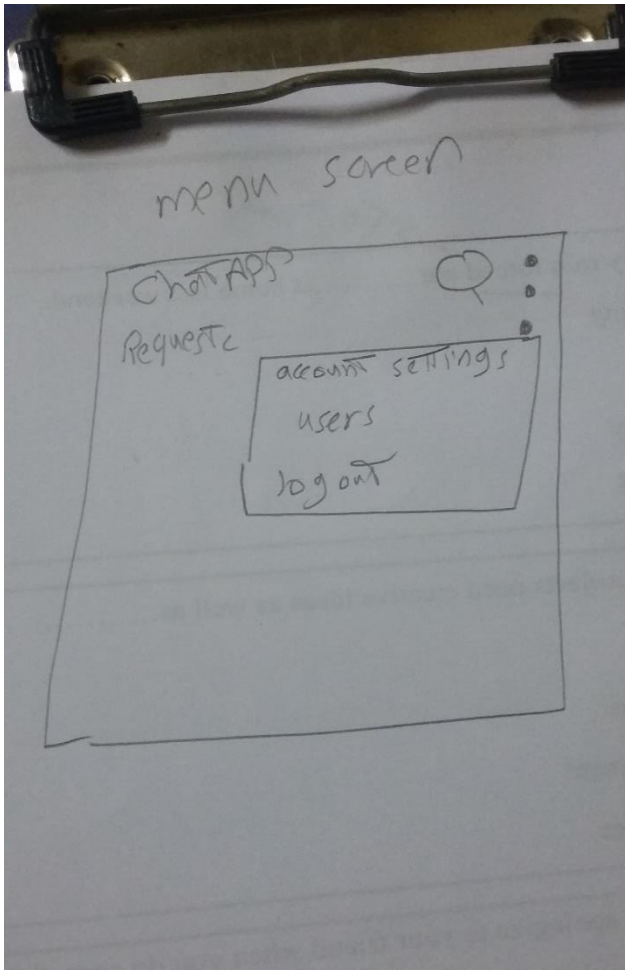

Reset password
login
reset password
Email

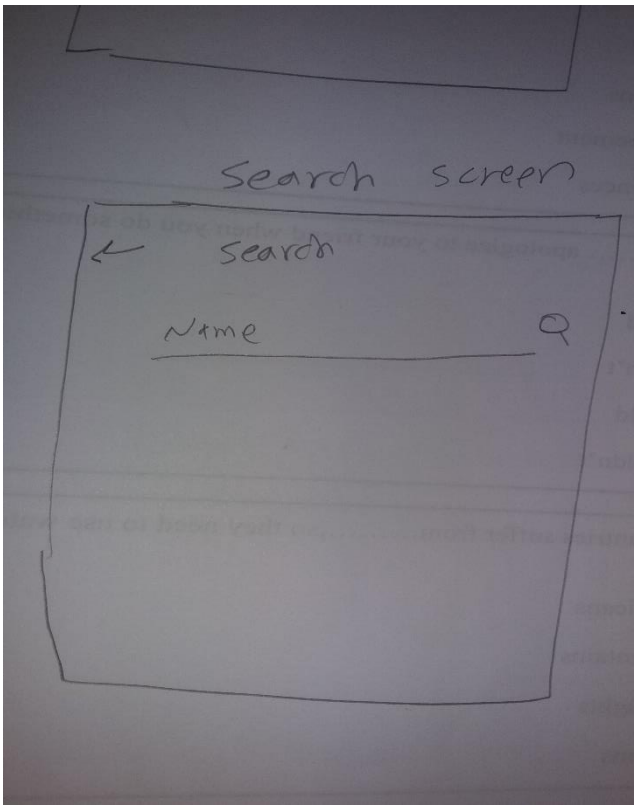

create account
signup
account Name
Email
password
submit

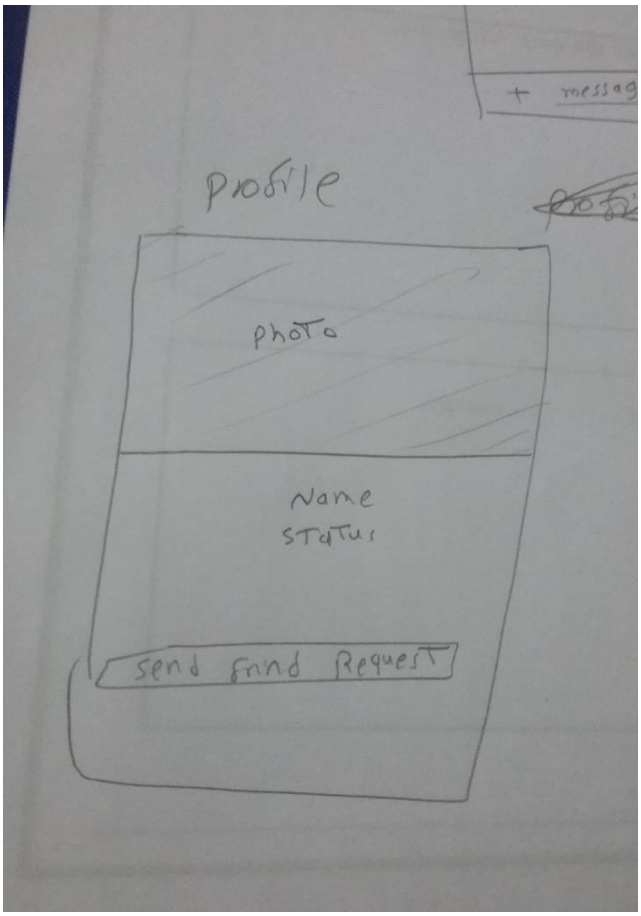

Request screen
ChatApp
Requests
Chats
Friends
Name
status
accept
Decline

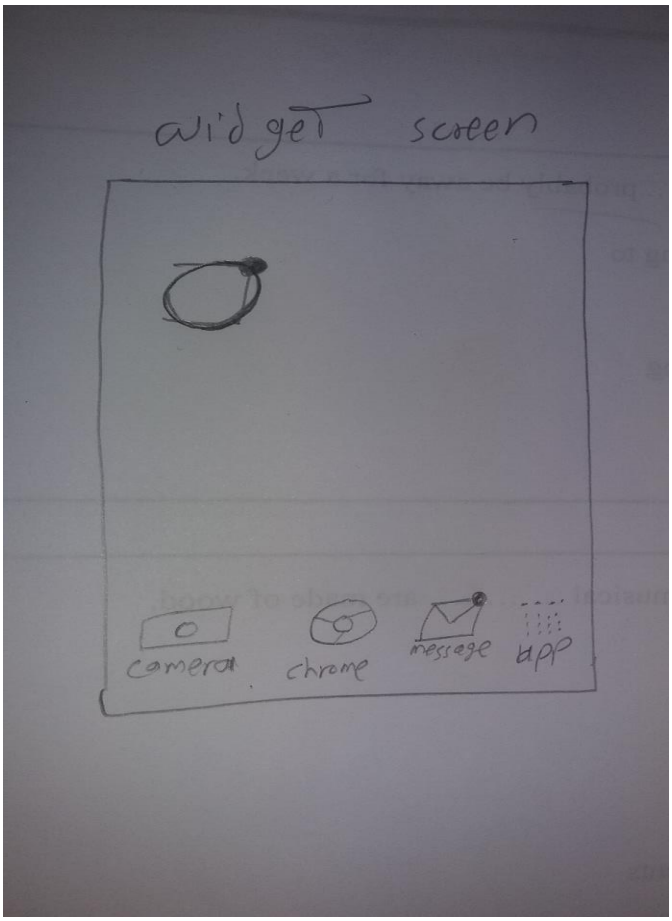

Chats screen
Requests
Chats
Friends
Name
text
Name
text
Name
Text

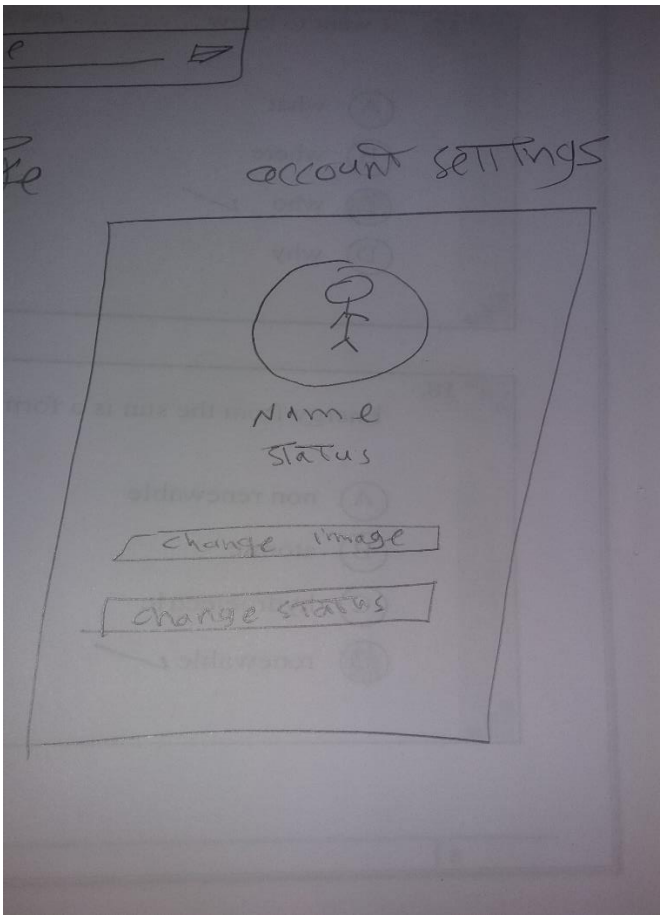

friends screen
Requests
chats
Friends
name
test
name
test


menu screen
Chat App
Requests
account settings
users
log out


search screen
search
Name


profile
photo
Name
status
send friend Request





Key Considerations

How will your app handle data persistence?

ChatApp use Firebase Realtime Database and Firebase Storage to store users data and files.

Describe any edge or corner cases in the UX.

- **Press Back Button from Main Activity after the login** – We have to be sure that after the user successfully logged into the App, he can't return to authentication screens pressing the back button.
- **Image PlaceHolders** – We have to consider loading times for pictures especially if heavy. So it is better to use an Image PlaceHolders to inform the user about loading and about contents.

- **Consider to implement an Image/Video Compressor** – Reducing the loading time for images is an important feature. Compressing them grant stability in database and performance in the app.
- **Offline Capabilities** – Firebase can be very powerful to manage offline status. The Database can be synchronized to specific DatabaseReferences. If something happens during an offline time, Firebase is able to keep it in memory and synch the updates as soon as the network comes back.
- **Chat Pagination** – Chat pagination can create some problems about duplicate messages when loading past messages. We have to implement a method to solve the problem.

Describe any libraries you'll be using and share your reasoning for including them.

- compileSdkVersion 27
- minSdkVersion 21
- targetSdkVersion 27
- build tool version 27.1.1
- gradle version 4.4
- android plugin version 3.1.4
- **Picasso '2.71828'**– Picasso is a good Library to manage images and very easy to use. With the support of Firebase and OkHTTP it can offer some peculiar features as offline image persistence.
- **MaterialEditText** – When we want to add some interesting Material features to editText this is a nice Library. Floating labels and Max/Min Characters are just few features that this library can offer.
- **OkHTTP '3.11.0'**– OkHttp perseveres when the network is unstable: it will silently recover from common connection problems.
- **Android-Image-Cropper '2.7.+'**– It's a powerful, customizable, optimized and simple image cropping library for Android. I think it can be useful on an app where users can upload very big images and choose to crop a detail to use as profile image.

- **Compressor '2.1.0'**– Compressor is a lightweight and powerful android image compression library that allows compressing large photos into smaller one with negligible loss in quality.
- **CircleImageView '2.2.0'**– A fast circular ImageView perfect for profile images.
- **FirebaseUI '3.3.1'**– Allows to quickly connect common UI elements to Firebase APIs.

Describe how you will implement Google Play Services or other external services.

ChatApp uses Google Play services as Firebase. The App implements the following modules:

- **FirebaseAuthentication** – to manage the user with an email and password registration and login.
- **Firebase Real-Time Database** – to store and fetch data inserted by users. Contains data about users from FirebaseAuthentication and Firebase Storage
- **Firebase Storage** – to store media files (images) inserted by users and provide a source link in the Firebase Real-Time Database.
- **Firebase Messaging** – to manage friendship request notifications
- **Firebase Invite** - Firebase Invites are an out-of-the-box solution for app referrals and sharing via email or SMS. To customize the invitation user experience, or to generate links programmatically, we use Firebase Dynamic Links.

Next Steps: Required Tasks

Task 1: Project Setup

Create a new Android project in Android Studio with empty activities and implement Google Play Services and Firebase.

- Configure Google Play Services.
- Implement Firebase Core.
- Implement Android support Libraries for Material Design.
    - FirebaseUI for Android.

- Android Support Library (Design).
- Android Support Library (constraint-Layout).
- MaterialEditText Library.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for HomeActivity.
- Build UI for MainActivity.
- Build UI for LoginActivity.
- Build UI for RegistrationActivity.

Task 3: Implement Register, Login, Logout functions

Now that we have implemented Firebase Authentication and created all authentications UIs, we can create the logic to register, login and logout a user in our app, and store data in the Firebase Real-Time Database. If User is not logged in instead of MainActivity he should be redirected to HomeActivity to proceed with Authentication.

- Implement Firebase Authentication in Gradle.
- Build UI for Toolbar.
- Register a new user with Firebase Authentication.
    - User Displayed Name.
    - User email.
    - User password.
- User Login.
    - User email.
    - User password.
    - Set the option to reset password.
    - Set “Remember me” to store and grant direct access for next accesses.
    - Consider Logout option.
- Run and Test.

Task 4: Implement UI for MainActivity

When the user successfully login in ChatApp, he is directed to MainActivity.

- Build UI for MainActivity.
    - Requires Toolbar and TabLayout.

Task 5: Implement MainActivity Fragments

In ChatApp MainActivity contain 3 tabs (Requests, Chats and Friends). We have to implement a fragment for each activity.

- Add TabLayout and ViewPager.
- Build UI for Tabs.
    - Build UI for Requests (Fragment).
    - Build UI for Chats (Fragment).
    - Build UI for Friends (Fragment).

Task 6: Implement a provisional Menu

Implement a provisional menu to edit Account Settings, to Logout access All Users account (just for test purpose).

- Add a Menu.
- Add Menu option: All Users.
- Add Menu option: Account Settings (Current User Profile).
- Add Menu option: Logout.

Task 7: Implement MainActivity

User, after successful login in ChatApp, pressing the back button must close the app and not navigate to authentication activity procedures. We also need to implement Menu and Permissions Request.

- Configure BackButton correctly.
- Add Menu Activities to MainActivity.

        - Logic to Account Settings.
        - Logic to All User.
        - Logic to Logout.
- Add Permissions Request function.
- Run and Test.

Task 8: Implement Account Settings Option

ChatApp consider 2 kind of users:

1. App User.
2. Other Users.

So we need to create 2 profile UIs depending on the user type. In these layouts we will insert users data and for do that we need to implement Firebase Real-Time Database.

- Build UI for Account Settings.
- Implement CircleImageView Library in the Gradle.
- Implement Firebase Real-Time Database in the Gradle.
- Implement Firebase Real-Time Database in RegistrationActivity to store user info to use on Account Settings.
- Run and Test.

Task 9: Implement Change Status and Change Profile Image in Account Settings

Provide the user with the tools to change some personal information such as Profile Image and Status message. We will use an Image Cropper Library to grant to the user the ability to crop the profile image to required size.

- Implement Picasso Library in the Gradle.
- Implement Android-Image-Cropper Library in the Gradle.
- Implement Firebase Storage in the Gradle.
- Implement functions Change Status.
- Implement functions Change Profile Image.
        - Implement Image-Cropper in Activity and Manifest.

    - Implement Firebase Storage in Activity.
- Implement a ProgressBar.
- Run and Test.

Task 10: Implement All Users Option

This options is temporary, just to manage data in Database structure creation and verify that all works fine. We will implement a Compressor Library to create profile image thumbnail to decrease image resolution.

- Build UI for all users.
- Build UI for all users item.
- Implement firebase-UI.
- Create users class (POJO).
- Implement Users ViewHolder extends RecyclerView.ViewHolder.
- Implement onStart method with FirebaseRecyclerAdapter.
- Implement Compressor Library in the Gradle.
- Implement Compressor in SettingsActivity to compress image retrieving profile picture and store it on Storage and DataBase.
- Implement a ProgressBar or ProgresDialog.
- Build UI for ProfileActivity.
- Implement onCLickListener on All Users Item.
- Run and Test.

Task 11: Implement User Profile

Clicking on a All Users Item from RecyclerView we will navigate to selected user Profile. Here we will implement the features to “Send Friend Request”,

“Cancel Friend Request”, “Accept Friend Request”, “Decline Friend Request” and “Unfriend User”

- Build UI for user profile.

- Implement firebase database.
- Implement a ProgressBar or ProgresDialog.
- Implement friend request methods.
    - Send friend request.
    - Cancel friend request.
    - Accept friend request.
    - Decline friend request.
    - Unfriend.
- Run and Test.

Task 12: Implement Offline Capabilities

Implement offline capabilities for Strings in our Firebase Database and, use OkHTTP and Picasso to expand this feature to images. In this way if the user is offline he can anyway access to app data.

- Create an application class.
- Set FirebaseDatabase persistence enabled in Application Class.
- Set DatabaseReference synch in SettingActivity.
- Adjust Manifest.
- Implement OkHTTP Library in the Gradle.
- Add Picasso Offline Capabilities in Application Class.
- Add Picasso NetworkPolicy.OFFLINE when we retrieve images URL from firebase database.
- Run and Test.

Task 13: Implement Firebase Notifications

Implement a notification system using Firebase Notifications to provide information about “Send Friend Request”. To do so we have to create a Firebase Function using Node.js and create a Notification Builder to grant the user to click the notification received and navigate to

the requesting user profile.

- Implement firebase notifications in Gradle.
- Test notification.
- Create function on Node.Js.
    - Users device tokenId are needed.
    - Retrieve and add device tokenId in user database on LoginActivity and RegisterActivity.
- Create FirebaseMessagingService class persistence enabled in application class and add it to manifest.
- Build a notification.
- Add an intent filter in manifest for ProfileActivity.
- Run and Test.

Task 14: Implement Friends Fragment

Now that we have developed a friendship request method, we can implement Friends Fragment to show a list of the user's friends.

- Implement Friends Fragment UI with RecyclerView.
- Use all users item UI as friend item.
- Create friends class (POJO).
- Implement friends ViewHolder extends RecyclerView.ViewHolder.
- Implement onStart method with FirebaseRecyclerAdapter.
- Implement onCLickListener on Friend Item.
- onClick we open an AlertDialog to let choose the user if he want "navigate to Friend Profile" or if he want "send a message to friend".
    - Implement intent open profile.
    - Implement intent open chat.
- Run and Test.

Task 15: Implement Presence System

Add a feature to show if the users are online or not.

- Implement Firebase Database and Firebase Authentication in Application Class.
- In Firebase Database Reference set a key inside “Users” for “online” and set it (true or false) according to OnDisconnect.
- In MainActivity set “online” to true onStart and to false in onStop. Do the same for every activity that need to notify the online status?
- Add online status image to All Users Items.
- Run and Test.

Task 16: Implement ChatActivity

In Task 14 we have implemented the AlertDialog to open ChatActivity.Now we have to create ChatActivity to finish that implementation.

- Build UI for ChatActivity.
    - Create a custom ActionBar to display UserName,LastSeen and profileImage.
    - Create a LinearLayout at the bottom to insert and send message.
    - Add a RecyclerView to show messages in Chat.
    - Create a single message list item UI.
    - Create drawable for message bubble.
- Create a class GetTime to calculate the time to show in lastSeen that extend application.
- Create message class (POJO).
- Create a MessageAdapter.
- Run and Test.

Task 17: Implement Send and Receive Messages

Now it’s time to send and receive messages.

- Implement onClickListener on send message button in activity bottom bar to launch sendMessage method.
- Get EditText message and send it to firebase database according the users (sender and receiver).

- Retrieve messages from Firebase database and show in the ChatActivity.
    - Create a method loadMessages().
    - Change chat bubbles according to the chat users.

Task 18: Implement Chat Pagination

Chat can be very long and so it's better to create a pagination of messages, so that just a few amount of them are shown. Anyway, implementing a swipeRefreshLayout, we can allow the user to load more past messages. Usually a new message is inserted as last item in the RecyclerView list. So it's a good solution add a top swipe to load past messages. Anyway the task has some problems because loading and reloading can create cloned messages during pagination.

- Add ActivityChat RecyclerView inside a swipeRefreshLayout.
- Implement a loadMoreMessages method to keep new sent messages on another method.
- Run and Test.

Task 19: Implement send Image Messages

Now it's time to send and receive image messages.

- Implement onClickListener on plus button in the ChatActivity bottom bar.
- Add an intent to open gallery.
- Retrieve selected image Uri and upload the image to firebase storage and add the image downloadUrl to firebase database.
- Add an empty image view inside message item list.
- Retrieve message type (text or image) to handle the correct method to populate the ChatActivity.

Task 20: Implement search user menu option

Implement a search option to find a registered user. Evaluating whether to show all app users

or just friends.

Task 21: Implement Chats Fragment

Populate the chats fragment.

- Implement chats fragment UI with RecyclerView.
- Use all users item UI as chat item.
- Create chats class (POJO).
- Implement Chats ViewHolder extends RecyclerView.ViewHolder.
- Implement onStart method with FirebaseRecyclerAdapter.
- Implement onCLickListener on chat item.
- onClick we open an AlertDialog to let choose the user if he want “navigate to User Profile” or if he want “send a message to friend”.
- Implement intent open profile.
- Implement intent open chat.
- Run and Test.

Task 22: Implement Request Fragment

Populate the request fragment.

- Implement request fragment UI with RecyclerView.
- Use all users item UI as request item.
- Create request class (POJO).
- Implement request ViewHolder extends RecyclerView.ViewHolder.
- Implement onStart method with FirebaseRecyclerAdapter.
- Implement onCLickListener on request item.
- onClick user navigate to requesting user profile.
- Implement intent open profile.
- Run and Test.

Task 23: Implement Home Screen Widget

Implement a home Screen widget to allow the user to add some important feature, such as a favorite chat with someone on the homepage of his device and keep it updated, or have a quick launch to that chat.

Task 24: Implement Land Responsive design

Some layout and activities could not respond as wished to device rotation. If necessary implement a land layout or enclose in a scrollview.