Family Chat a chat app tutorial using MIT CloudDB database by SteveJG

Here is *Family Chat*, an App Inventor 2 app and tutorial using the MIT CloudDB to connect several users in a live chat.   This tutorial shows a way to build a Chat app using CloudDB.

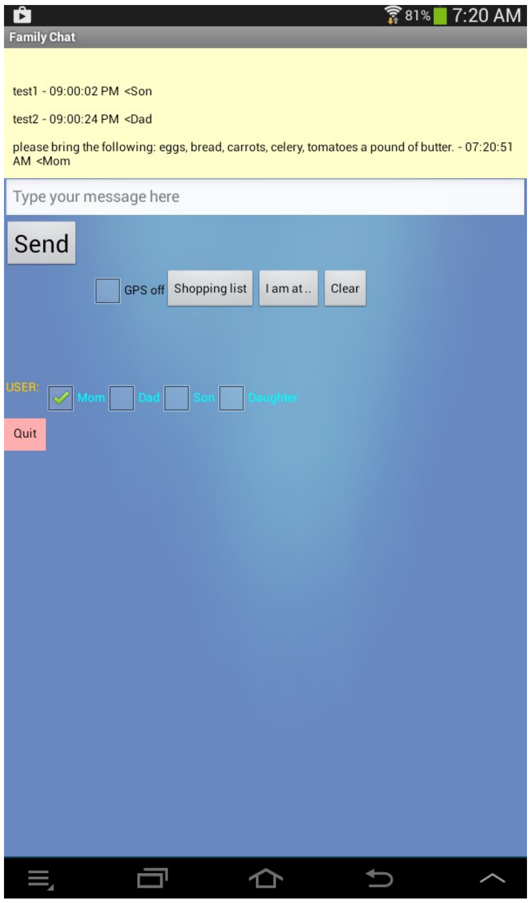
Family Chat uses **Responsive Sizing** and is designed on a Tablet. The blocks might need minor adjustments to run properly (scale) on your device.

You must test this on your Android device. CloudDB apps presently will not run in the stock emulator that comes with App Inventor 2. This is an Intermediate to advanced level Project.

Users can only share the same chat room if the app is compiled and a copy of the apk is placed on each user's Android. All users must have a copy of the master apk on the phone/tablet they use to communicate with each other. You cannot see each other's chats unless all users install the same apk.

Family Chat

Family Chat allows you and your family or friends to have a live chat provided all members of your circle have a copy of this app on their phone/tablets. You do not need a personal Redis database account to use Cloud with this sharing app, the app uses the server MIT provides for testing. If you want/require privacy, a personal Redis DB account may be required.



test1 - 09:00:02 PM <Son

test2 - 09:00:24 PM <Dad

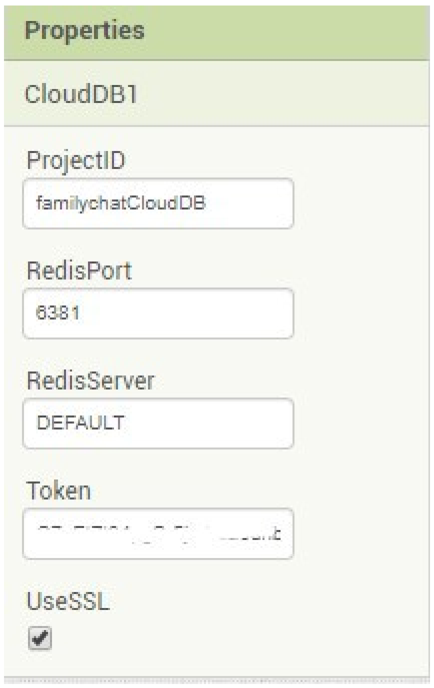
please bring the following: eggs, bread, carrots, celery, tomatoes a pound of butter. - 07:20:51 AM <Mom

Family Chat displays up to 10 comments at one time on the screen. The oldest message is replaced by the newest chat as the chat room fills up.

A time stamp and a provision for an auto ‘signature’ for family members is included (use the check box to select your signature (Mom,Dad etc. in this example ... change these options to Bill, Mary, John etc. with your own coding modification). What USER you check is saved in a TinyDB so each user will have their own signature. An example ‘canned’ message about groceries shows a methodology to add shortcuts to your version of the app; modify and enhance this feature if you require the functionality. If the GPS box is checked, the app provides a street address (if available in Google's database) . When checked, you can include use if you use the **I am at** button (and an address is available in the Google database). The **Clear** button clears the input field; that simple. **SEND**, sends your chat message to everyone who has a copy of your app.

Install compiled copies of the app on various devices, connect with a WIFI to the Internet,and the devices using *Family Chat* can engage in a live chat. The phones/tablets share the default database located in the DEFAULT RedisServer. At MIT.

When you test this app, use the default server. When you load the aia, the compiler will automatically assign the Token When you are ready to finalize your version, you probably will need to change to your own Redis server.



Properties

CloudDB1

ProjectID

familychatCloudDB

RedisPort

6381

RedisServer

DEFAULT

Token

[redacted]

UseSSL

✓

The MIT sponsored service may not be a secure database. Anyone who has the code can join the chat if they connect using the RedisPort, RedisServer and Token used in your app (see MIT's documentation). For that reason, you may want to password protect your app (add appropriate coding not described here) and provide your own ProjectBucket name prior to compiling and using the app.

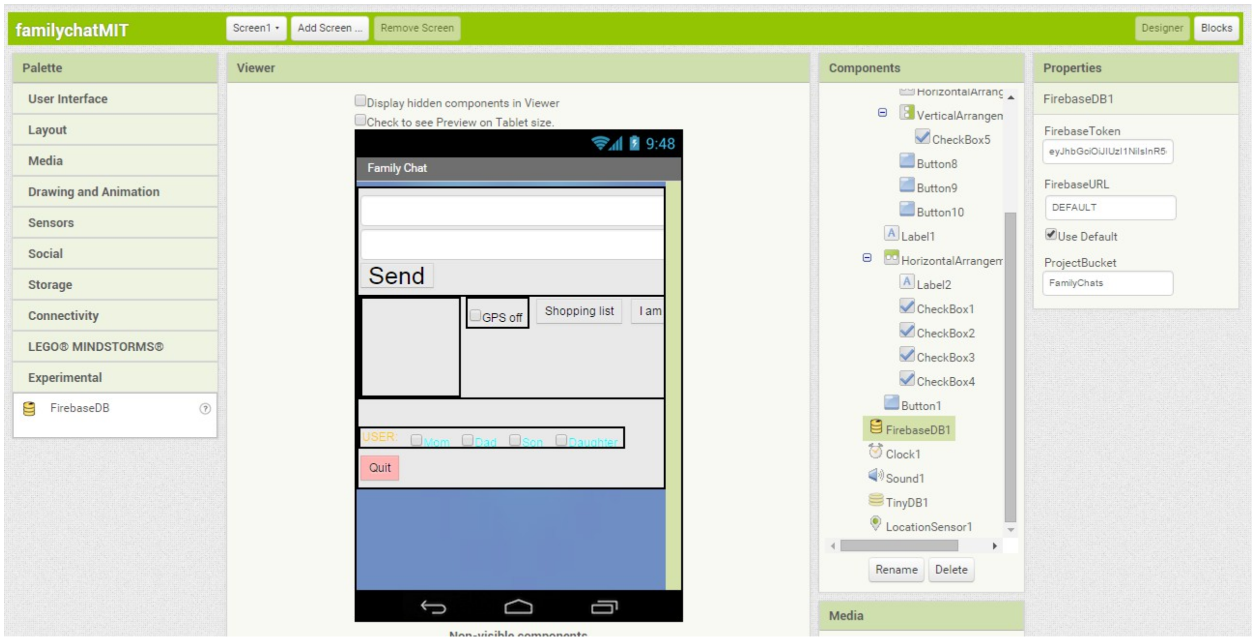
.

Testing:

After you build the app, test it before you share copies of your apk. When you send a message on 'your' app using the Send button, the message should show in the chat room (the yellow TextBox area) provided your device is connected to the Web using WIFI.   If the message displays properly after you post using the Send button then close the app and then restart the app to check if the message was stored. You should see displayed the previous chat in the chat room when you restarted the app.   If the conversation shows the last message, the app is working properly. If you do not see what you sent using Send, there is probably a coding error and or typo. in your version of *Family Chat*.

If the chat shows on the chat room display, you are ready to compile the app. Compile and send to and share the master apk with each person you want to use *Family Chat*. Test the app perhaps by sending perhaps from your tablet to your phone or to a friend's Android. The database will be updated even if one of the chat circle does not have their app turned on. If someone does not have the app on when your chat message is sent; any user will receive the message when they later turn on their copy of the app. Be aware this version of the app **only 'saves' the last 10 messages**.

The Designer Screen



FirebaseDB1

FirebaseToken: eyJhbGciOiJIUzI1NiIsInR5

FirebaseURL: DEFAULT

☑ Use Default

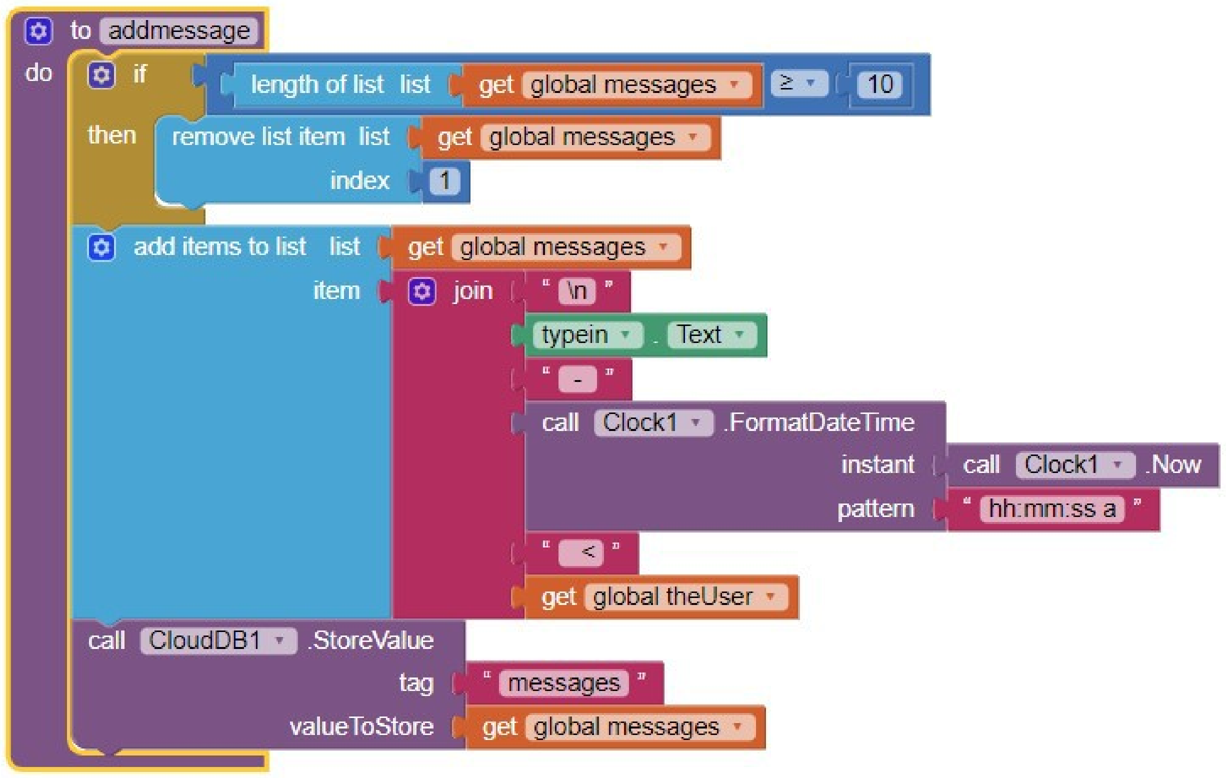
ProjectBucket: FamilyChats

The ProjectID is set to famillychatCloudDB; set that to whatever you want (MyChat, TestChat, etc.)

This app works as of the release of App Inventor 2 version 182 released early 2020. An account sponsored by MIT is configured for the MIT sponsored service in the app. *Family Chat* uses that service. You do not have to use that service for your account (the MIT documentation explains how you can use your own account).

The Blocks

AddMessage Blocks

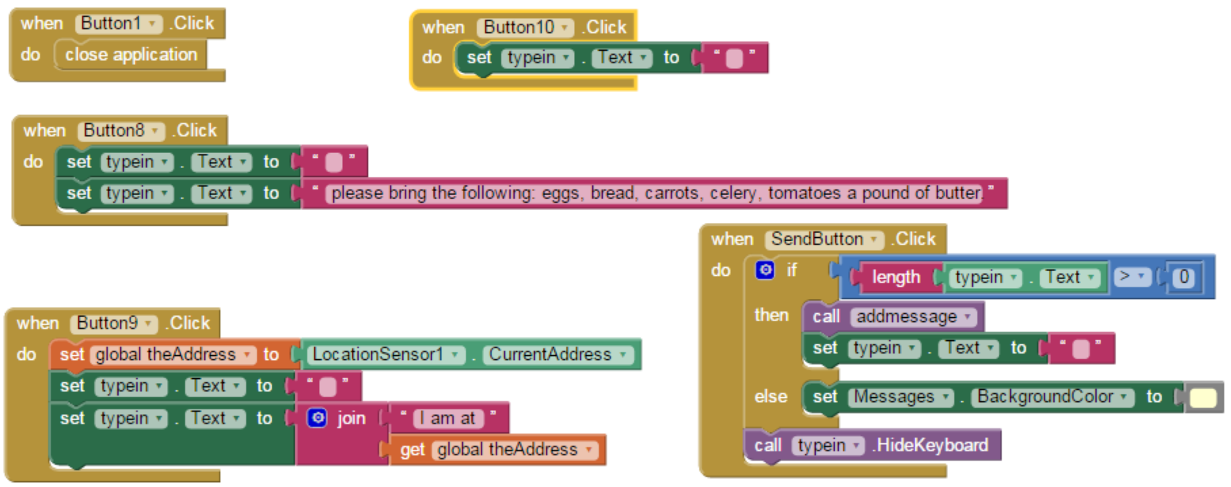


```
to addmessage
do  if  length of list list  get global messages  ≥  10
    then  remove list item  list  get global messages
                            index  1
    add items to list  list  get global messages
                       item  join  " \n "
                                   typein . Text
                                   " - "
                                   call Clock1 .FormatDateTime
                                               instant  call Clock1 .Now
                                               pattern  " hh:mm:ss a "
                                   " < "
                                   get global theUser
    call CloudDB1 .StoreValue
                  tag  " messages "
                  valueToStore  get global messages
```

This adds a message including a time stamp and the user ‘signature’ (saved in theUser variable).

The Buttons

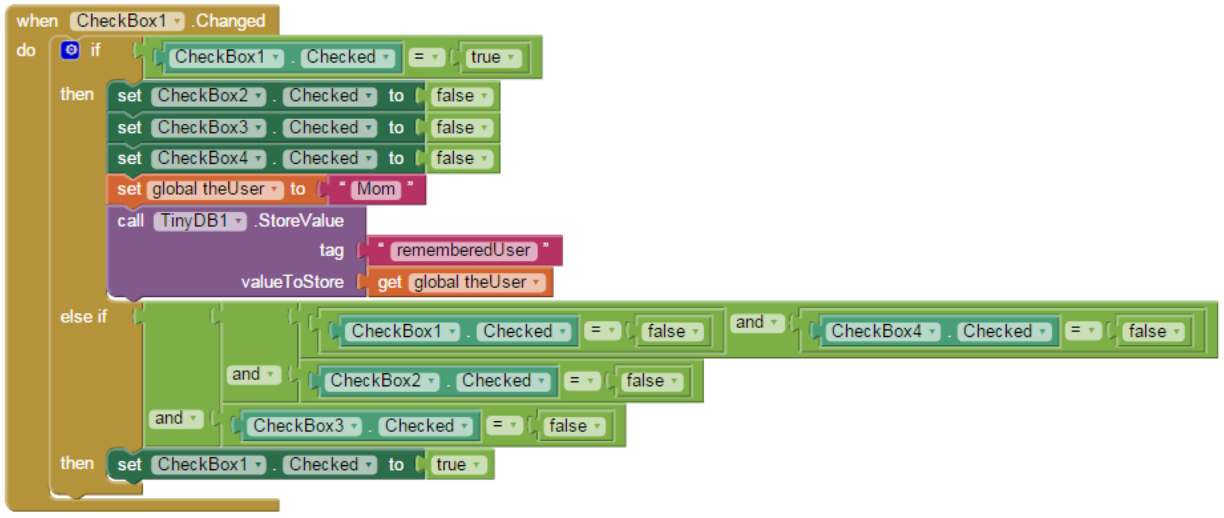
Button1 closes the app.
Button10 clears the text type in fields.
Button8 is an example ‘canned’ message.
Button9 sends the address of the user to the type in field using GPS or a WIFI geolocation.
SendButton actually sends your message to the Family Chat.



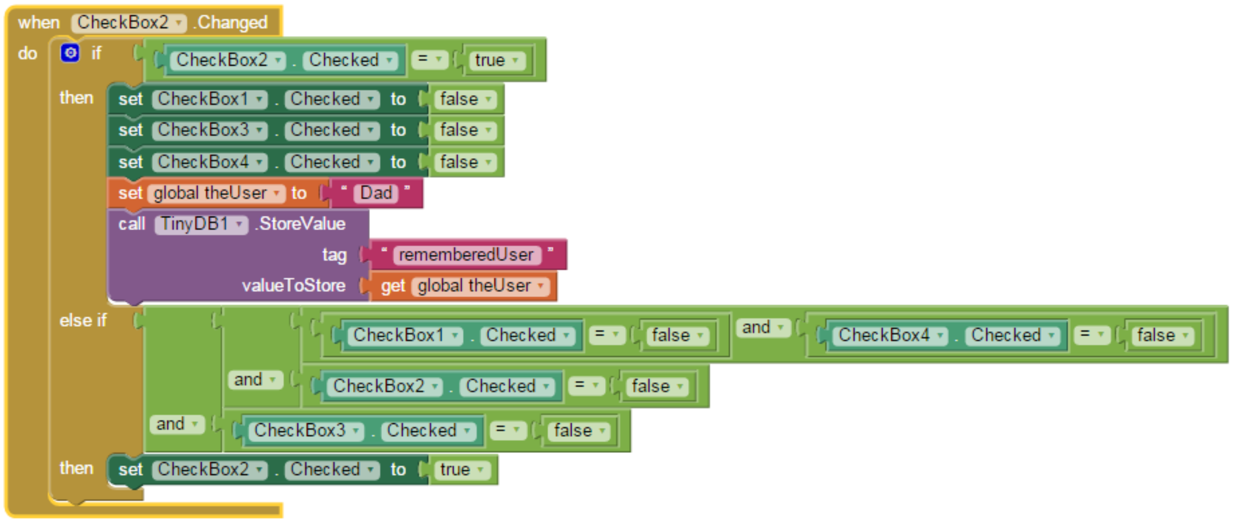
```
when Button1 .Click
do  close application

when Button10 .Click
do  set typein . Text to " "

when Button8 .Click
do  set typein . Text to " "
    set typein . Text to " please bring the following: eggs, bread, carrots, celery, tomatoes a pound of butter "

when SendButton .Click
do  if    length typein . Text > 0
    then  call addmessage
          set typein . Text to " "
    else  set Messages . BackgroundColor to
    call typein .HideKeyboard

when Button9 .Click
do  set global theAddress to LocationSensor1 . CurrentAddress
    set typein . Text to " "
    set typein . Text to join " I am at "
                              get global theAddress
```

The Checkboxes

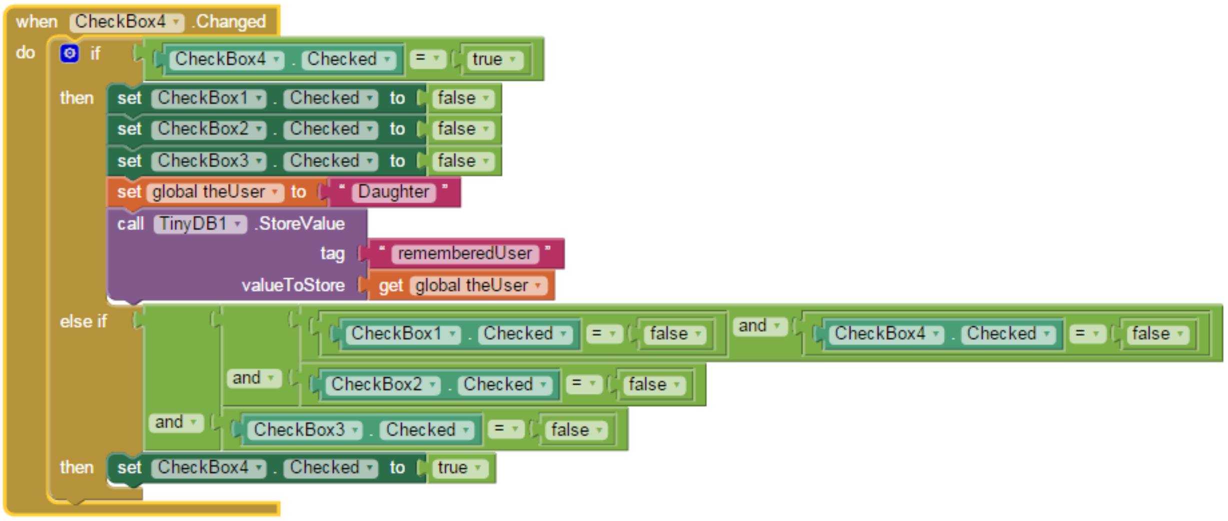
The first four check boxes store and remember the user of the Family Chat app on individual devices. The if routines select the current ‘signature’ of a chat user.



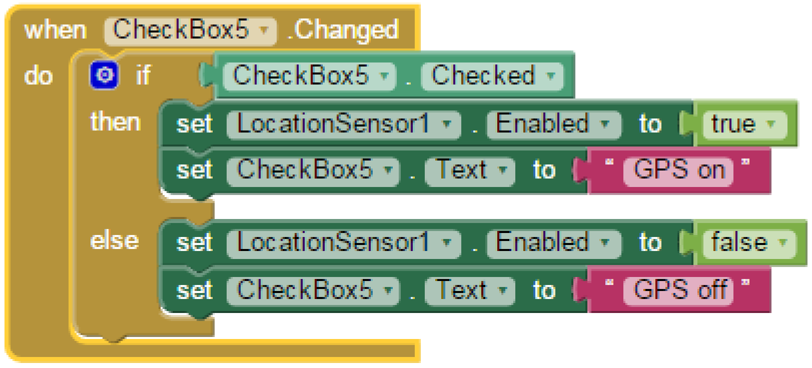
```
when CheckBox1 .Changed
do  if      CheckBox1 . Checked = true
    then    set CheckBox2 . Checked to false
            set CheckBox3 . Checked to false
            set CheckBox4 . Checked to false
            set global theUser to " Mom "
            call TinyDB1 .StoreValue
                                 tag " rememberedUser "
                        valueToStore get global theUser
    else if CheckBox1 . Checked = false and CheckBox4 . Checked = false
              and CheckBox2 . Checked = false
              and CheckBox3 . Checked = false
    then    set CheckBox1 . Checked to true
```



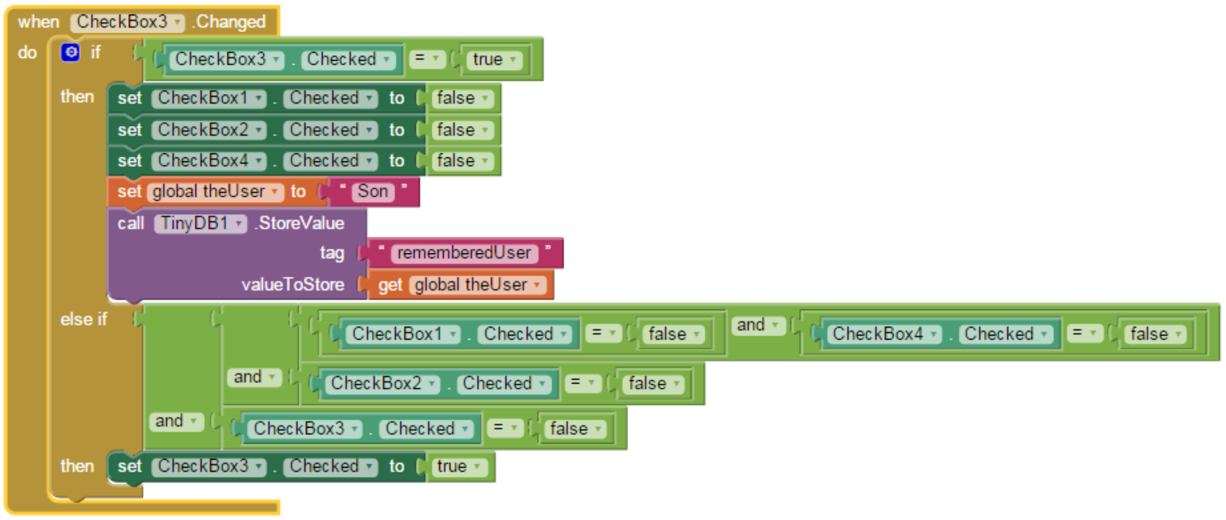
```
when CheckBox2 .Changed
do  if    CheckBox2 . Checked  =  true
    then  set CheckBox1 . Checked to  false
          set CheckBox3 . Checked to  false
          set CheckBox4 . Checked to  false
          set global theUser to  " Dad "
          call TinyDB1 .StoreValue
                           tag  " rememberedUser "
                  valueToStore  get global theUser
    else if         CheckBox1 . Checked  =  false   and   CheckBox4 . Checked  =  false
              and   CheckBox2 . Checked  =  false
          and   CheckBox3 . Checked  =  false
    then  set CheckBox2 . Checked to  true
```



```
when CheckBox4 .Changed
do  if   CheckBox4 . Checked = true
    then set CheckBox1 . Checked to false
         set CheckBox2 . Checked to false
         set CheckBox3 . Checked to false
         set global theUser to " Daughter "
         call TinyDB1 .StoreValue
                          tag " rememberedUser "
                 valueToStore get global theUser
    else if  CheckBox1 . Checked = false  and  CheckBox4 . Checked = false
             and  CheckBox2 . Checked = false
             and  CheckBox3 . Checked = false
    then set CheckBox4 . Checked to true
```



```
when CheckBox5 .Changed
do  if   CheckBox5 . Checked
    then set LocationSensor1 . Enabled to true
         set CheckBox5 . Text to " GPS on "
    else set LocationSensor1 . Enabled to false
         set CheckBox5 . Text to " GPS off "
```

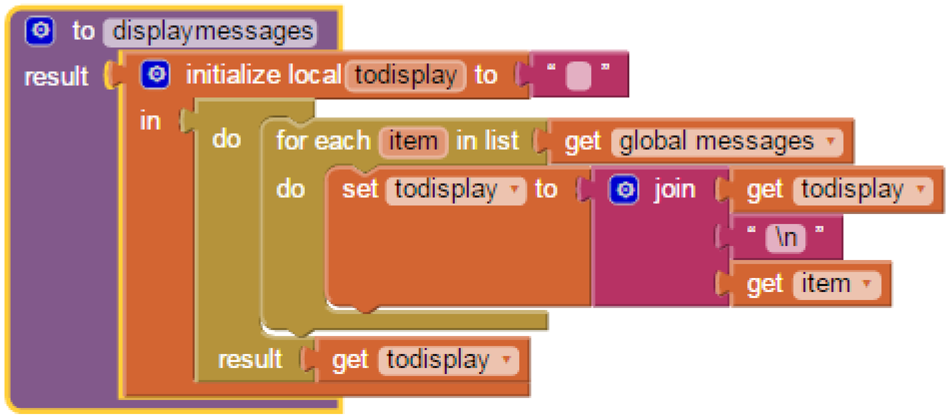


```
when CheckBox3 .Changed
do  if   CheckBox3 . Checked = true
    then set CheckBox1 . Checked to false
         set CheckBox2 . Checked to false
         set CheckBox4 . Checked to false
         set global theUser to " Son "
         call TinyDB1 .StoreValue
                          tag " rememberedUser "
                 valueToStore get global theUser
    else if  CheckBox1 . Checked = false  and  CheckBox4 . Checked = false
             and  CheckBox2 . Checked = false
             and  CheckBox3 . Checked = false
    then set CheckBox3 . Checked to true
```

Checkbox5 controls the GPS (turns it on and off).

Display Messages

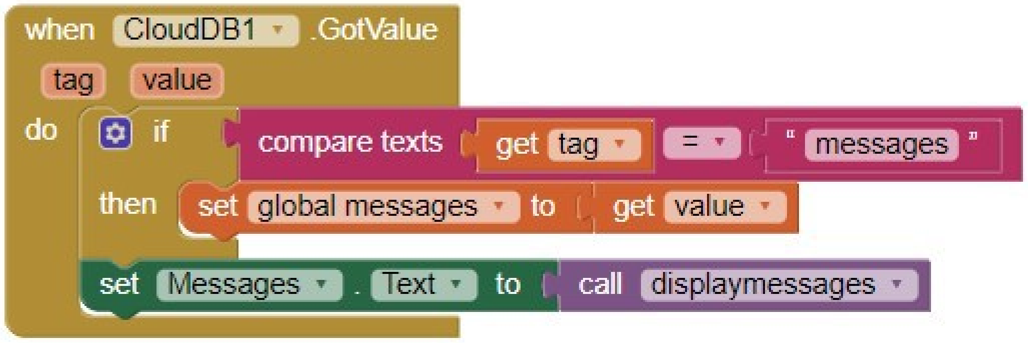
This is the procedure to display the chat messages.



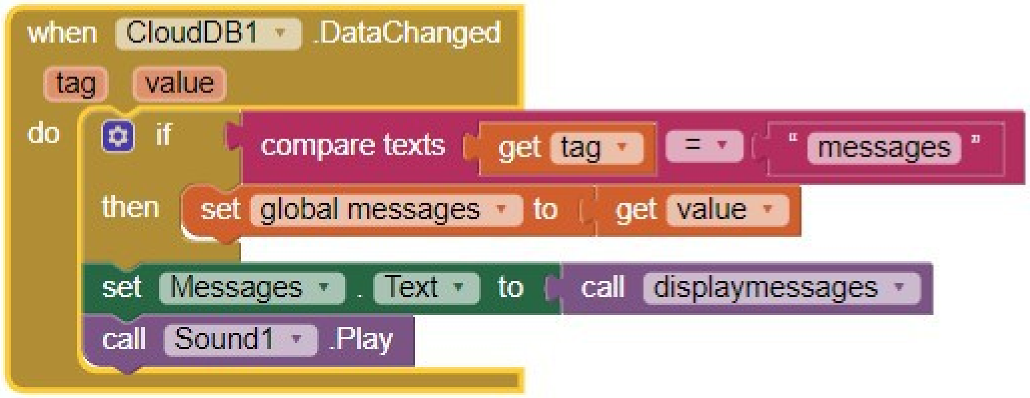
```
to displaymessages
result  initialize local todisplay to " "
        in  do  for each item in list  get global messages
                do  set todisplay to  join  get todisplay
                                            " \n "
                                            get item
            result  get todisplay
```

The CloudDB Control

Communicate with the CloudDB.



```
when CloudDB1 .GotValue
  tag  value
do  if  compare texts  get tag  =  " messages "
    then  set global messages to  get value
    set Messages . Text to  call displaymessages
```

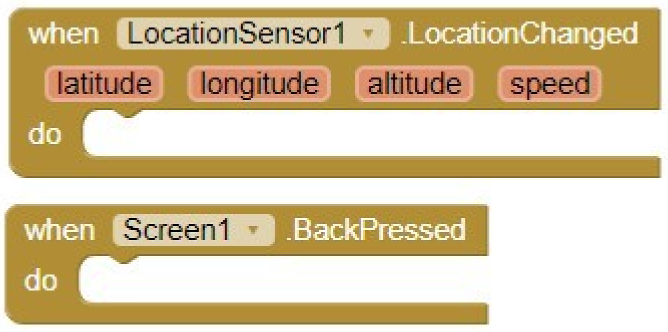


```
when CloudDB1 .DataChanged
  tag  value
do  if  compare texts  get tag  =  " messages "
    then  set global messages to  get value
    set Messages . Text to  call displaymessages
    call Sound1 .Play
```

Miscellaneous Blocks

The LocationSensor is required to post location information about the current user.

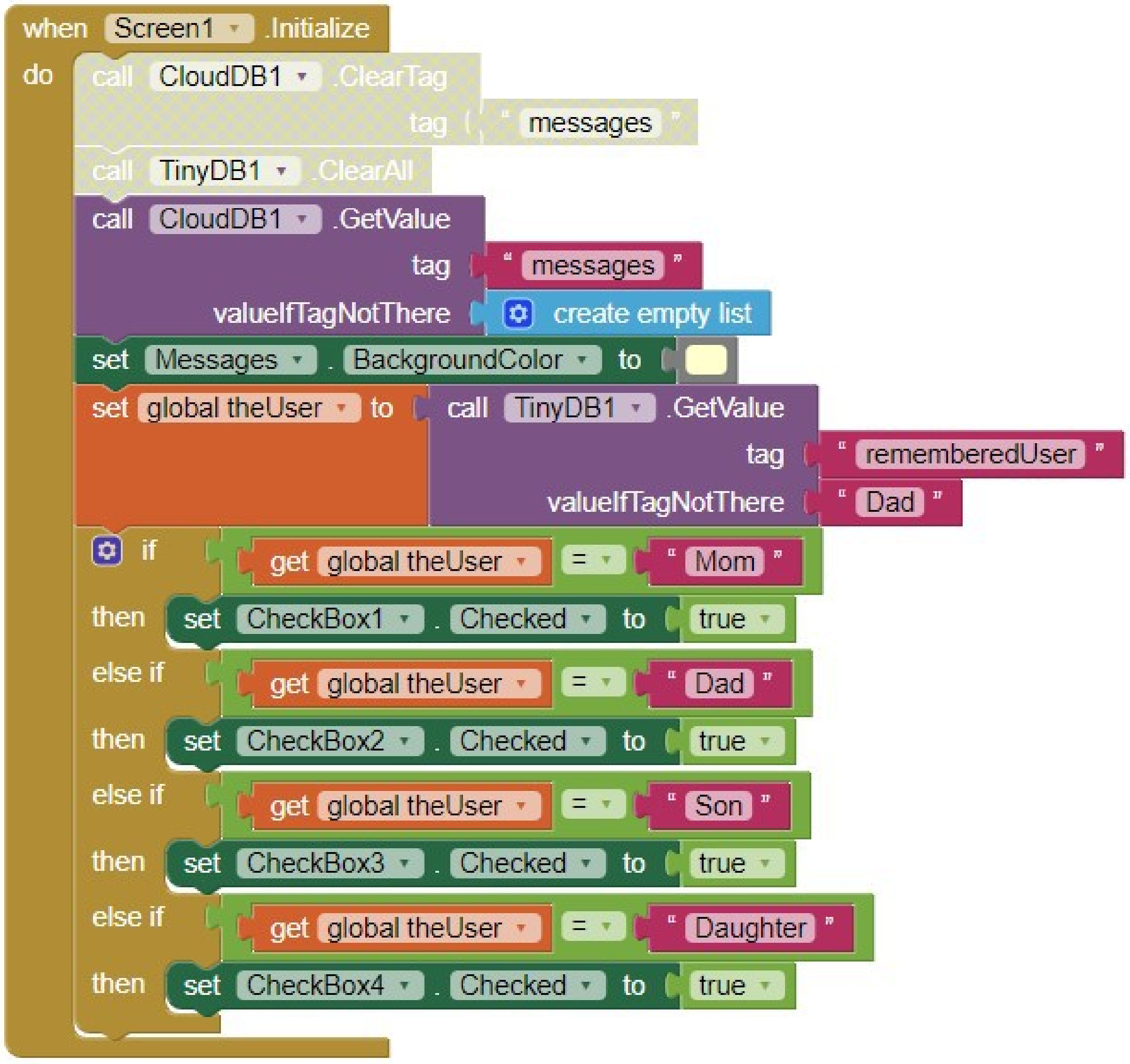
The Backpressed control is provided (empty) to prevent users from inadvertently terminating the app by pressing the virtual back button on the device.



```
when LocationSensor1 .LocationChanged
  latitude  longitude  altitude  speed
do

when Screen1 .BackPressed
do
```

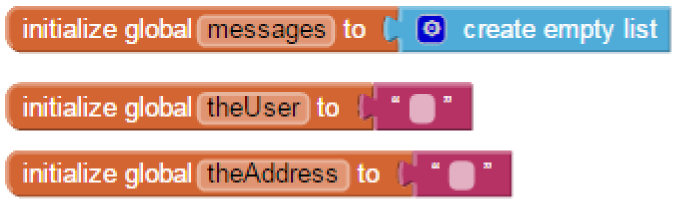
Screen1.Initialize

Provide the initial status for the app.



```
when Screen1 .Initialize
do  call CloudDB1 .ClearTag
                       tag " messages "
    call TinyDB1 .ClearAll
    call CloudDB1 .GetValue
                       tag " messages "
         valueIfTagNotThere create empty list
    set Messages . BackgroundColor to
    set global theUser to call TinyDB1 .GetValue
                                        tag " rememberedUser "
                            valueIfTagNotThere " Dad "
    if      get global theUser = " Mom "
    then    set CheckBox1 . Checked to true
    else if get global theUser = " Dad "
    then    set CheckBox2 . Checked to true
    else if get global theUser = " Son "
    then    set CheckBox3 . Checked to true
    else if get global theUser = " Daughter "
    then    set CheckBox4 . Checked to true
```

The Global Variables


```
initialize global messages to   create empty list
initialize global theUser to  " "
initialize global theAddress to  " "
```

The aia File

A Project aia file is attached.

Important Facts

Have fun with the coding for personal use. Use the algorithms and ideas in your own app and enjoy coding.