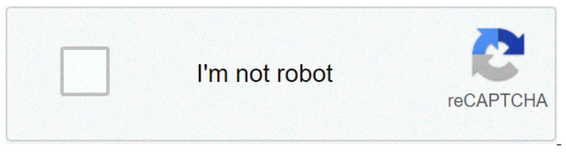

Continue

How to delete user data on android

How to delete system user data on android. How to delete data usage on android. How to delete data on android. How to delete data usage history on android.

[{ "Type": "thumbs down", "id": "missingTheInformationINeed", "label": "Missing I need information"}, { "type": "thumb down", "id": "tooComplicatedTooManySteps", "label": "too complicated / too many steps"}, { "type": "thumb down", "id": "outOfDate", "label": "out of date"}, { "type": "thumbs down", "id": "samplesCodeIssue", "label": "question Samples / code"}, { "type": "thumbs down", "id": "otherDown", "label" "Other"}] [{ "type": "thumbs-up", "id": "easyToUnderstand", "label": "easy to understand"}, { "type": "thumbs-up", "id "" solvedMyProblem "," label ":" Did this fix the problem "}, {" type ":" thumbs-up "," id ":" otherUp "," label ":" Other "}] To help help users control their data, the team created a library that simplifies Firebase two common user data processes: clearData: deletes data from a user's specific services Firebase (currently real-time database, FireStore, and Storage) when eliminates their account via the Firebase authentication. ExportData: Save a copy of a user's data from Firebase services in a JSON string, and loads of Cloud Storage so you can easily download it yourself. Read on to find out how to add features to your application, or jump right into the code at Firebase privacy of users GitHub repo. Note: The clearData ExportData functions and are a convenient way to manage data organized into a single user ID. If you use more complicated data structures, for example, the responses of nested messages under someone else's ID, you may need more advanced models for light and export functions. The clearData ExportData and functions in the library are implemented as cloud functions that operate on data in real-time database, Cloud Storage, and Cloud Firestore. Adding the functions in your application is a three-step process: Change the file user_privacy.json the library to reflect your app database in real time, Cloud FireStore, and Cloud Storage scheme. Distribute clearData and ExportData as cloud functions. Implement triggers for the functions of your app. Fix the clearData data storage rules. Edit the file user_privacy.json the library To start, clone or download the privacy of users Firebase GitHub repo. When you have done this, open functions / user_privacy.json in a text editor. The json file has a number of customizable routes that clearData and ExportData functions use to find your app data in real-time database, cloud FireStore, and Cloud Storage. If your application uses only one or two of these services, start eliminating the JSON objects associated with unused services. With the missing items, you can begin to replace the placeholder values the remaining services at the actual data structures are using your app. Add the base paths of real-time data to user data to customize user_privacy.json for example real-time database of your app, replace the placeholder strings in the list of "database" with the actual paths to the user data: .. . "database": { "clear": ["/ users / UID_VARIABLE", // Replace it with your "/ administrators / UID_VARIABLE" // actual paths RTDB], "export": ["/ users / UID_VARIABLE" / / Replace with your "/ administrators / UID_VARIABLE" // paths RTDB real]}, ... If you only want to add one of the functions for your application, you can delete the object of the other function, instead of filling it with your data. Add cloud Firestore objects that contain the data user to customize user_privacy.json eg cloud Firestore your app, replacing the list of placeholder objects "FireStore" with the current cloud Firestore which contain user data: ... "Firstore": {"Clear": [{"Collection": "Users", "Doc": "field" "uid_variable" ,: "name"}, {"Collection" : "Users", "DOC": "UID_Variable"}, // Replace with your {"Collection": "Administrators", "DOC": "UID_Variable"} // Actual Firstore Routes], "Export": [{{ "Collection": "users", "doc": "uid_variable", "field": "name"}, {"collection": "users", "doc": "uid_variable"}, // replace with your { "Collection": "Administrators", "DOC": "UID_Variable"} // Actual Firstore Paths]}, ... If you want to add only one of the gods For your application, you can delete the other function of the other function, instead of filling it with your data. Add the Cloud Storage bucket and the file name with data to the user to customize user_privacy.json for example cloud storage of your app, replace the placeholder storage bucket and the file name in "storage" with actual values :. .. "Storage": {"clear": [// Replace with your actual storage data ["clear-export.appspot.com", "uid_variable / sample_data.json"], ["clear-exportappspot.com" , "Uid_variable"]], "export": [// Replace with real data storage ["clear-export.appspot.com", "uid_variable / sample_data.json"]]}, ... if you want to add Only one of the features for your application can be deleted the other function, instead of filling it with your data. If you are not yet familiar with the cloud functions, read how to use them in the Get introductory guide functions. Once you are comfortable with the cloud functions, add the ClearData and Export Doctors to the project: Copy your user_data.json customized to the functions directory. Copy the code from Index.js of the Library Privacy of Users to Index.js of your project. If you do not use Cleardata, omit the Cleardata, ClearDatabasedata, Clearfirestorata and ClearstoragedData functions. If you do not use exportData, omit the exportData, exportDatabasedata, exportFirestoratoData and exportStorDataData functions. Distribute the functions. Each function requires a different trigger: Cleardata: activated when a user deletes his account through authentication. ExportData: activated by an HTTP request. To activate a Cleardata event, you need to use a method of authentication from. If you haven't done it yet, add authentication for your application: iOS, Android, or Web. Then add a way to recall the authentication SDK delete method for the platform: firebaseAuth.user.Delete {error in the case of error Let = error {print ("User error error: (error)")}} FirebaseAuth .getCurrentuser () .elimina (); Firebase.auth (). Currentuser.Delete (). Capture (Function (error) {if (error.code === 'Auth / Requires-recent-Login') {window.alert ('enters-in and try again.'); firebase.auth () Signout ();. }}); To implement an exportata trigger, add a button or link for your application that recalls the function via an HTTP request. To learn more about functions by invoking HTTP into call functions via HTTP requests. Request details: Type: Post URL: https: // Noi-Central1- project-id .cloudfunctions.net / exportata body: current_user's_uid Call up the function directly into Firebase Hosting If your application is a web application hosted on Firebase Hosting, you can recall the Cleardata function through a rewrite voice in the Firebase.json file of the site: "host": {"rewrites": [{" source ":" / exportata "," Function ":" exportData "}]} Protect the exportata data with storage rules to keep user-exported data users, add cloud storage rules that limit access to the user export. Visit storage in the Firebase console. Open the Rules tab. Paste the following rule, then public: Firebase.storage service {match / b / {} bucket / o {match / exportData {// only allows user access that required the export game / {uid} {Allow Reading, writing: if request.auth.uid == uid} match / {uid} / {path = **} {allow you to read, write: if request.auth.uid == uid}} // other application rules ...} Note: verifies the overlap of cloud storage rules. Lost extensive cloud storage access rules overwrite the most restrictive ones. Check that you don't Of a pre-existing rule that could invalidate your new rule. This "Clear Data" option is seriously broken for the case of use of the initial screen widgets that could have shared preferences and / or some data stored in the local cache. It is also completely against the documentation for developers who say: ". You can save files directly into the internal memory of the device by default, the files saved in the internal memory are private â €