Requirements in terms of Data Storage and Processing

1. Data to be stored

a. User signed-in data (user ID, name,phone number{mandatory},Device ID ,User type : shopkeeper,consumer etc.)

b. Data corresponding to every scan genuine or fake with the following details .

i. Scan ID(unique hash value)
ii. Scan result (G/F)
iii. Product info(all the details in QR code)
iv. Key index(vital in case of missing product info)
v. Time and date
vi. Geo-location
vii. Label no.
viii. Scanning time till getting the result Genuine or fake
ix. Number of times QR decoded
x. Number of times inactive screen came in process of scanning
xi. DiffCount

Special case of fake:

Data field by the user on the report link
Product details*
Shop details
User comments
User contact number

c. Product info data customized according to a brand stored against their public key. It will two database in actual implementation one for JSON key-value pair and other multimedia data. JSON key-value will have references of multimedia data storage.

d. Image corresponding to genuine or fake result,In case of fake one more image: Binary image of tag(as we

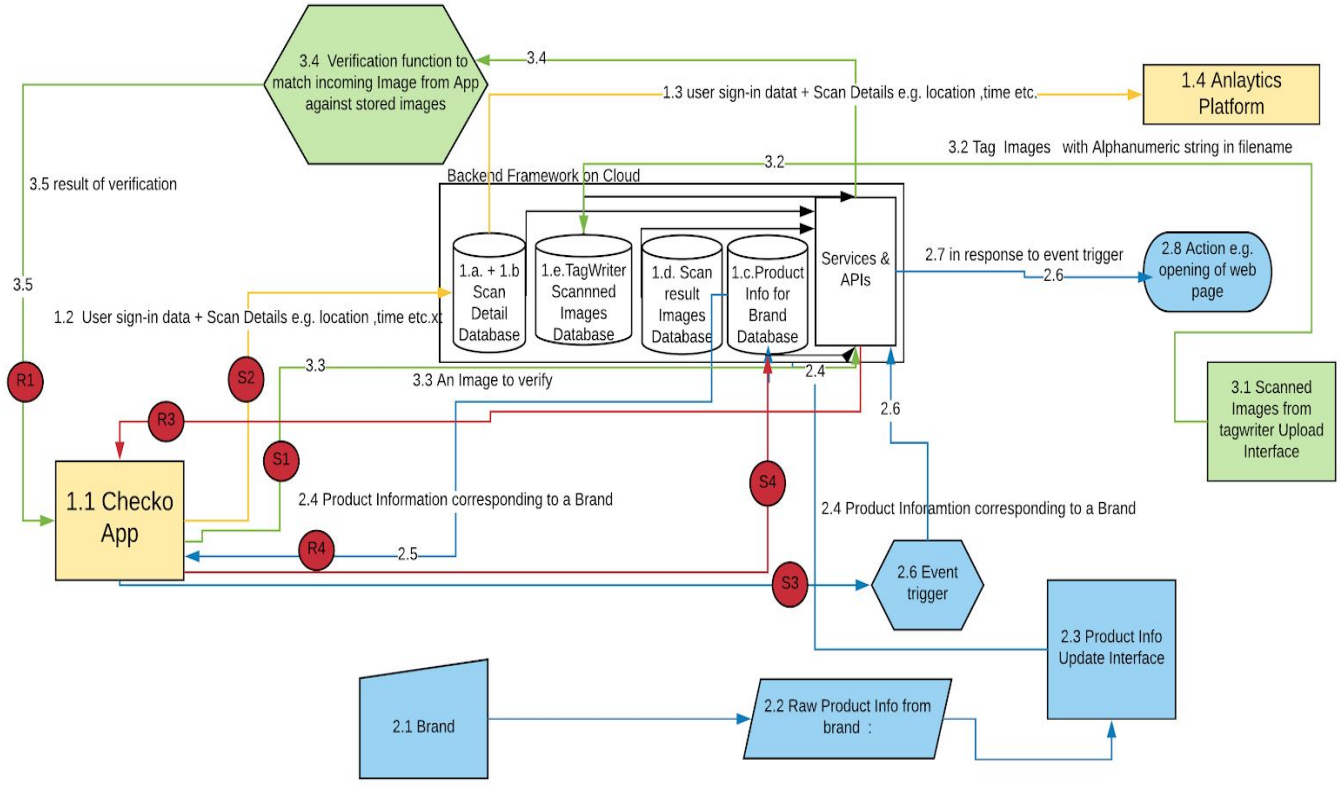
call it in code: pentagon failure) **note** - stored against scan id ,simultaneously with 1.b in order to bind(map) this image to the data stored in 1.b . It is Done because the data types in 1.b and 1.d are different.

e. Scanned images generated for tagwriter having filenames with Alphanumeric String (multiple reads but only single source of writing) (internal analytics)

2.Data to be Processed

a. Data in step 1.b. Is to be used for analytics and may need some preprocessing for streaming to analytics platform.
b. Data in step 1.c. Can different data types and may require some further based on user action e.g. opening of a web page etc. (accessed against public key given to brand and product info in QR code if more than one)
c. Images in step 1.e. would be used to cross match against image sent by user's app. (Online Verification)
d. Coupling of image in step 1.d. To data in step 1.b.

Architecture Diagram



Architecture Diagram Explanation

A. There are 3 major backend flows described in the diagram above . We'll go through them one by one starting from flow 1.

B. Flow 1 deals with data described in reference 1.a + 1.b in the document .
C. 1.2 is the step of sending the data from user’s phone to realtime database in firebase .
D. In 1.3 the same data stored in firebase is streamed to analytics platform at 1.4
E. Flow 2 deals with more Product Info data which we show to the user on genuine result screen to access more information about the product scanned. This option only works at user app if the phone is connected to internet(not in offline) .
F. 2.1 represents the Brand owner of the products in the market having checko authentication.
G. 2.2 is the product info corresponding to a product which brands wants to show the end user (their customer).
H. 2.3 is an interface available to us as well as to the brand to describe and define the product info they want to show. It would be a software interface uploading the data at the our firebase database.
I. 2.4 depicts uploading the product info data to database with reference 1.c .
J. S4 is the event of product info request from genuine result screen of App.
K. 2.5 depicts sending of JSON key-value pair data to the App also having references to multimedia data if any .
L. 2.6 represents event of user clicking on any reference to multimedia data stored for a product info . In response to this event an API will fetch the multimedia from the reference .
M. Depending on the data type i.e. image,video,web link etc required action will happen at user’s phone. Action may or may not be inside the App e.g. downloading of image
N. 2.7 & 2.8 represents the above point M.
O. Flow 3 deals with the scenario of online image verification.
P. 3.1 depicts a software interface uploading the 3D tag scanned images using which the Cryptocode has been generated for offline verification.

Q. These images would be stored in datastore with reference 1.e. Indexed on the basis of alphanumeric key representing unique id for each tag image.
R. Each image is ~ 100 KB
S. 3.2 represents storing of these images in the datastore.
T. 3.3 represents 3D tag image coming from app for online verification along with alphanumeric key.
U. This image is received by the online verification service running at backend and the same service searches images in 1.e. Using the alphanumeric key and on successful retrieval of image matches it with the app from the App. All this is represented in 3.4
V. 3.5 shows passing of the verification result of 3.4 to the App.
W. R1,R3,S1,S2,S3,S4 are having the references in data flow diagram provided for App .

Technical Implementation Overview

1. Data would be saved and retrieved from cloud database using APIs(data contract) mostly RESTful ones.
2. There is two types of data in broad category : a) text based data mainly JSON key-value pairs b) multimedia data mainly in form of images.
3. For these two types of data mentioned above in point 2 we would maintain separate data stores(type of database) for them.
4. For images we would do their indexing and searching on the basis of their alphanumeric string in filename.
5. For image matching we would to host an run a service on the cloud itself.
6. The data obtained from user through the app would be streamed to a analytics platform. (a Data Pipeline has to be build)
7. For product info event trigger based function have to be defined on cloud

Technology Stack Options for Implementation

1. Firebase + google cloud services : Details are available at the following link https://cloud.google.com/solutions/mobile/mobile-app-backend-services
2. AWS services :https://d1.awsstatic.com/aws-answers/aws-mobile-app-backend.pdf

Suggested Technology Stack For Implementation

The option 1 is suggested for the implementation . In option 1 itself there are 5 options namely :

a) Firebase
b) Firebase & App Engine Standard environment
c) Firebase & App Engine flexible environment
b) App Engine Standard environment & endpoints
b) Compute Engine & REST/gRPC

In this option b & c is recommended for our use case . We would be using both the options because native library (openCV) based services can only be run on option C. And option b is default for all the other services & API as it is cheap as well easier to

configure for scaling. For better analysis below is their comparative matrix .

Feature	Firebase	Firebase & App Engine standard environment	Firebase & App Engine flexible environment	App Engine standard environment & Endpoints	Compute Engine & REST/gRPC
Automatic capacity scaling	✓	✓	✓	✓	If you configure an autoscaler.
Automatic real-time data synchronization	✓	✓	✓		
Automatic server maintenance	✓	✓	✓	✓	
Backend logic		✓	✓	✓	✓
Call native binaries, write to the file system, or make other system calls.			✓		✓
Data storage	✓	✓	✓	If you add other Cloud Platform services	If you add other Cloud Platform services
File storage	✓	✓	✓	✓ with Cloud Storage	✓ with Cloud Storage
Easy user authentication	✓	✓	✓	OAuth 2.0	
Language support for mobile backend services	N/A	See the App Engine documentation		See the Cloud Endpoints Frameworks documentation	Any
Messages and notifications, such as push notifications	✓	✓	✓	✓ with Cloud Messaging	✓ with Cloud Messaging
Platform support	iOS, Android, Web	iOS, Android, Web	iOS, Android, Web	iOS, Android, Web	iOS, Android, Web