Centrric.

Centrric Innovations Pvt. Ltd
CIN.: U72900KL2021PTC068397

Device Management - Case Study

Remote Device Management for Apps/Platforms

The Scenario

1. The mobile app can run multiple user accounts (I can switch between multiple accounts)
2. Each user account could be an admin of the business profile (A high privileged profile just like Facebook Pages)
3. Multiple admins for a single business profile
4. Multiple devices for all the users
5. Notification needs to be sent to each device for each profile
6. Business profile notification also needs to be sent to the business profile admins

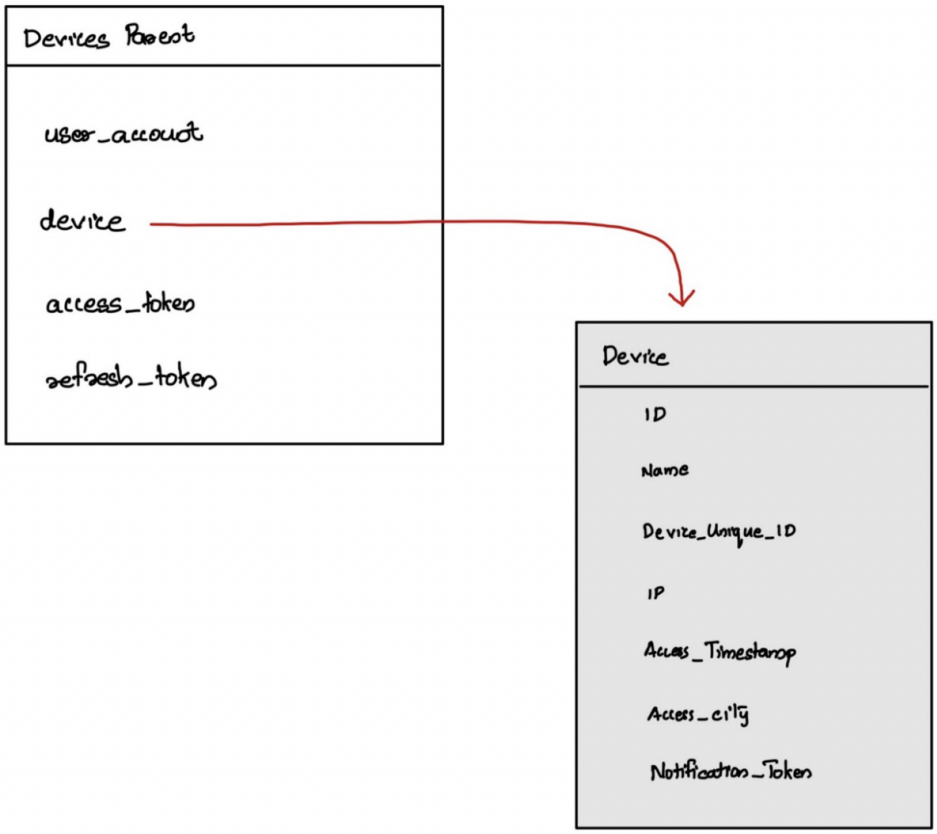
Requirement

1. Need to save device information corresponding to each user account
2. A device could have multiple user accounts in the app. That flow also needs to be covered.
3. Efficient function to send notifications (Personal profiles and business profiles also multiple devices)
4. Manage access and refresh tokens corresponding to each user account and device

Solutions

For the multiple problems we faced, we figured out the solution by introducing a full-fledge Device Management Module

Saving Devices (Backend)



Devices - Models

Each device should generate a unique ID, which can be done using various packages on React/Flutter/any front end framework.
This unique id is passed along with all other device information to the backend.

The above table shows that a single user account could hold any number of devices just by creating another DeviceParent object.
The access token, refresh token and fcm token generated are also saved.

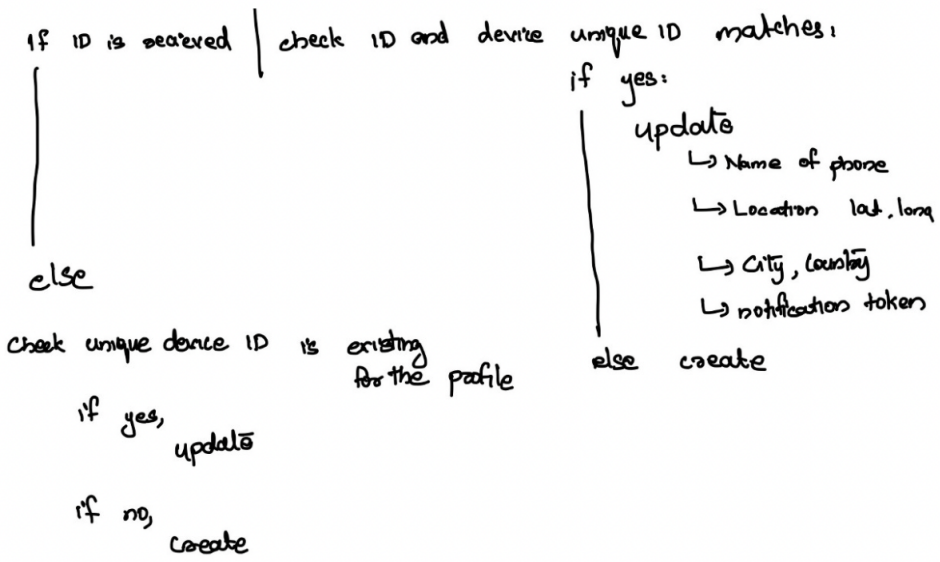
Device Registration (Backend perspective)

Device registration function should recieve the following arguements

1. Device Unique ID
2. Device Name
3. IP (Received from the HTTP request)
4. City, Country (Optional)
5. FCM Token / Any push notification service token
6. Access token & Refresh token

If the device unique ID already exists in the devices DB, then update the data, else create a new device in the DB with the received data

If the device object ID in the backend is also stored in the front end, then the checking for device-existing-or-not will be less complex. We have implemented it, but its completely depending on the scenario. It should work without the object ID as well. (Note that the device object id and device unique ID are not same. Object ID is the DB level ID)



My rough note while thinking the architecture. This explains the same that I have written above.

Device Registration (Frontend perspective)

Now the architecture is ready. We need to save/register the device to the backend.

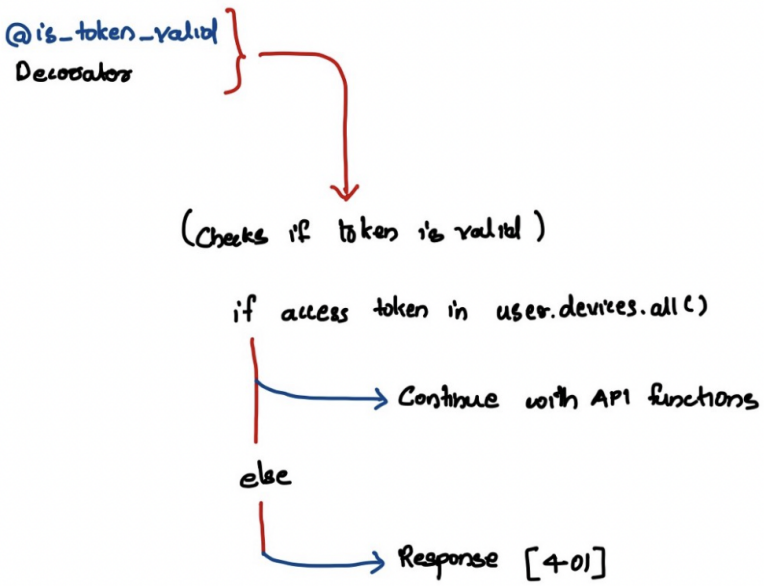
For our case, the access token is generated each time the app opens.
So for us, there were 3 cases

1. Signup
2. Generating access token using username and password
3. Refreshing the access tokens

We enabled device registration for all these cases.
The device info along with generated access token, refresh token, fcm token and device unique id are used to register the device to the backend.
Now it's ensured that, the device will be registered in all cases.

Token validation decorator

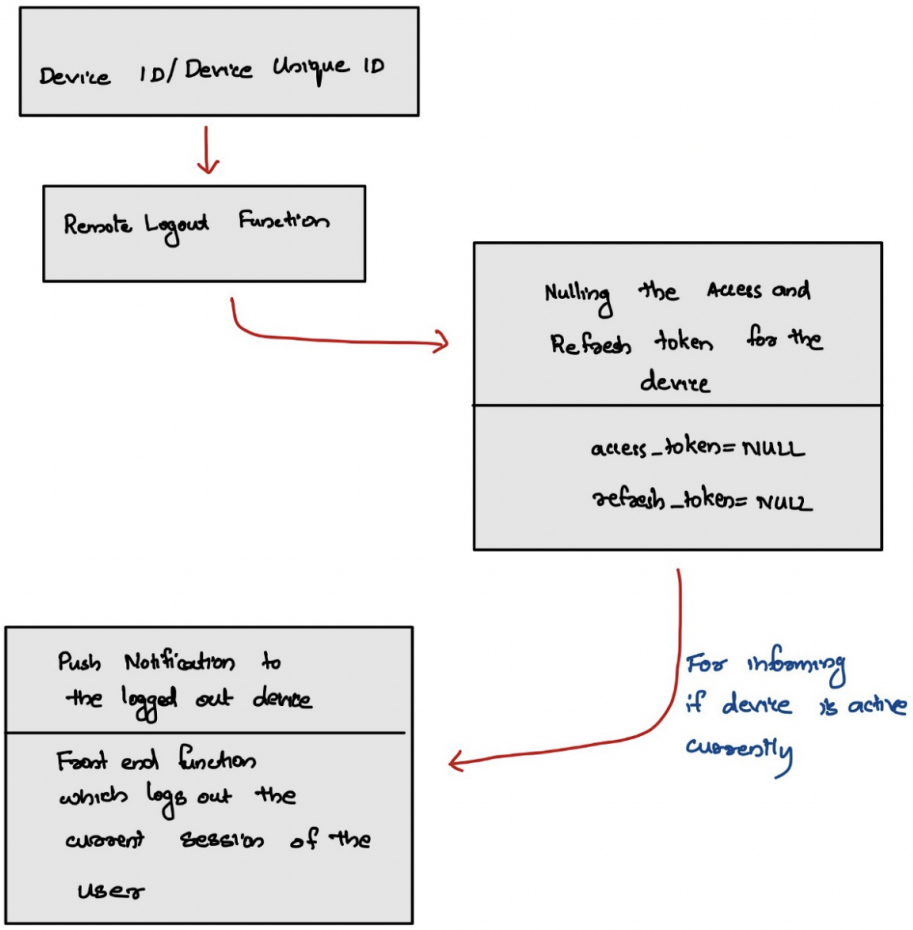
We built a decorator for all the APIs to validate if the access token received in the header of the HTTP request.
If the access token is available in the devices module connected with the user account, then the token is marked as valid, as the sequence goes to the rest of the functions of the API.
If the access token is invalid, a 401 the API will respond with a 401 response



Another planning time note. This explains the above textual explanation.

Logout remotely from a device

Created an API to remotely logout the user account from a device.
The function receives only the device unique ID.
The function nullifies the access and refresh token in the device module for the user account. The token validator decorator on all apis will validate and respond 401 for all the requests from the device immediately.
And if the device is active currently, we send a silent push notification to the app not logout the user account which will in turn log out the user account on the device immediately.



Conclusion

The above architecture totally works for our scenario. This may be different depending on the conditions and the business requirements of the app that you are working on.

Also, there is room for improvisation in this architecture. If you have any idea to improve my flow, please comment and let's all contribute to each other.