React Native App Specification

NOTE:

- There is no design for the app. You need to use React Native Elements or NativeBase (or nothing) as your UI toolkit. You cannot use anything else.
- You will not be judged for the UI or the UX. The only expectation is that we are able to carry out the actions described below
- In instances where you need to display a message, do not show it momentarily. You will have to show it like a “status” - visible at all times.
- The app below mentions the Proof API. This is still a WIP api and for now, we only have the swagger specification to share. You can refer to it when calling the api. In your implementation, you could provide a configuration for the api endpoint - we will go with https://webhook.site/ or equivalent for now to manually inspect your apis to verify them. You can provide a mock api here as well (in nodejs only - mock api that does the bare minimum for the mobile app is enough) - strongly recommended.
- You need to support both Android and iOS devices. Only mobile devices are in scope for now.

Login (Major Requirement)

- The app, when launched, needs to present a button to the user to “Login using Topcoder”. This button, when tapped, should take the user to Topcoder’s login page through Auth0.
- You will use ‘https://topcoder-dev.auth0.com/’ for your domain and “1NpddnMe5M3r04W71F95wDZZTNubWl5u” for your client id. You can test login using lazybaer / appirio123 for login.
- After login, the app will get the userId of the logged in user. Remember this for Session Pairing below
- After login is a success, the user then sees a dashboard where there is a button that allows the user to Scan a QR Code.
- The user should also have the option to logout, which logs out the user and which takes the user back to the landing page (Login using Topcoder button page)

QR Code Scanner (Major Requirement)

- Once logged in, the user has the option of scanning a QR Code. The button is named “Pair a Session”
- When the button is tapped, the user should be able to scan a QR Code (You can provide a dummy QR code to scan for now)
- The QR Code that the user scans will contain a “sessionId” - this is basically a token that is generated from Topcoder’s VSCode extension which the app needs to scan
- The sessionId will be a uuid
- Once you retrieve the sessionId, stop scanning and proceed to session pairing (see below)

Session Pairing with Proofs API (Major Requirement)

- So, after login, the app would have the user id.
- And once the user would have scanned the QR Code, the app would have the sessionId.
- Using the Session id, the app will then make a GET request to Proofs api to get details about the session using the id (GET /v5/proofs/sessions/{sessionId})
- The response will be of the form (besides the audit fields):
    - userId
    - status
- Verify that the status is “Pairing”
- If the status is not “Pairing”, but is “Active”, then display message “Device has already paired. No further action needed”.
- If the status is not “Pairing”, but is a value other than “Active”, then display message “Session pairing expired or invalid. To re-initiate pairing, logout and login once again to Topcoder through VSCode”
- If the status is “Pairing”, then proceed to make a PATCH request to the Proofs api at /v5/proofs/sessions/{sessionId} with the status field changed to “Active”.
- If the API succeeds, display the message “Session Pairing: Active”.
- Allow the user to “detach” the pairing, in which case make another PATCH request to the proofs api at /v5/proofs/sessions/{sessionId} with the status of “Closed” and forget the sessionId. The user will have to scan the QR Code again.
- Note that “display the message” above is displaying the message in the screen itself, and not through a popup or a modal. The last message displayed should be visible at all times.

Send Location to Proofs API (Minor Requirement)

- Now that the user has logged in and has “paired” successfully (session pairing task above), the app will now periodically (configurable) send the location (longitude and latitude) of the mobile device to the Proofs api
- It will do a POST request to /v5/proofs/proofEvents and pass the session id (that it received from the QR Code scanner and which it used to successfully pair), the device id (determine the device id, which will uniquely identify the device and remain the same for all sessions), the proof type (which will be “Geolocation”) and the geo location itself (string of the form “longitude, latitude”).
- On the UI, display a message “Monitoring Location: Active” to let the user know that the location is being sent periodically.
- If the api fails, ignore the failure. Try again at the next interval.

Periodically poll the Proofs API (Minor Requirement)

- After session pairing is a success, periodically (configurable) poll the Proofs api at /v5/proofs/sessions/{sessionId} and verify that the status in the response (not the http status, but the status attribute in the response body) is still “Active”. In this period, the user continues to see the message “Session Pairing: Active” in the UI

- If the status is “Closed”, then display the message “Session Pairing: Closed”.
- If the status is anything other than “Active” or “Closed”, display the message “Session pairing expired or invalid. To re-initiate pairing, logout and login once again to Topcoder through VSCode”