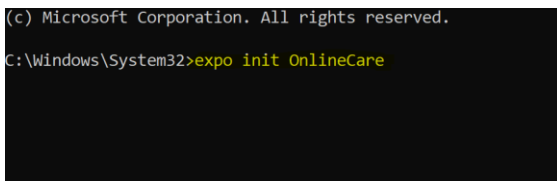Md Rayhan (He/Him)

Launching smartphone app in 15 minutes

The primary objective of the smartphone app is to restrict users from recording the screen during playback of paid lecture videos. The company has been facing challenges as some users have been using screen recording tools to copy and distribute the paid lecture videos illegally. To counter this issue and protect the intellectual property of their content creators, Online Learning Care decides to launch a smartphone app for their website.

```
3
4   //import React in our code
5   import React from 'react';
6
7   //import all the components we are going to use
8   import { Text, View, SafeAreaView } from 'react-native';
9
10  import { WebView } from 'react-native-webview';
11
12  import { usePreventScreenCapture } from 'expo-screen-capture';
13
14  const App = () => {
15
16    usePreventScreenCapture();
17
18    return (
19      <SafeAreaView style={{ flex: 1 }}>
20        <WebView
21          source={{ uri: 'https://online.udvash-unmesh.com/Account/Login' }}
22          style={{ flex: 1, marginTop: 20 }}
23        />
24      </SafeAreaView>
25    );
26  };
27
28  export default App;
29
```

Picture Caption: Roughly about ten lines of code that wraps the website as WebView app and prevents users from capturing the app screen (don't hessitate to reach out if you need help for your brand)

Here, we develop a WebView application for the Online Learning Care and check the capability of blocking screen capture.

```
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>expo init OnlineCare
```

Picture Caption: The very first command "expo init <AppName>"

Thereafter, we make sure we install "react-native-webview" & "expo-screen-capture".

It was imperative to decorate the app with the company's branding. Figma community has a template for expo app icon and flash screen which we used.

It is important to make sure the icon and splash screen's path is included in the "app.json" file.

Picture Caption: Overriting the "./assets/icon.png"

The command "expo start –tunnel" helps to test the app with expo-go client. There are options to inject JavaScript and modify the UI components e.g., hiding the footer. For now, we stick to the basic one. Let's ship it.
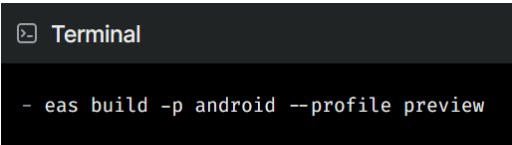
```
⊡ Terminal

- eas build --platform android
```

Picture Caption: building the installer for android platform.

By default, this command produces ".aab" installer for the app store.

```
5    "build": {
6      "development": {
7        "developmentClient": true,
8        "distribution": "internal"
9      },
10     "preview": {
11       "android": {
12         "buildType": "apk"
13       }
14     },
15     "production": {}
16   },
17   "submit": {
18     "production": {}
19   }
```

Picture Caption: eas.json file buildType

In case you are interested in ".apk" file, make sure to include buildType as "apk".

```
⊡ Terminal

- eas build -p android --profile preview
```

Picture Caption: build command to create apk

After running the above command, the ".apk" was ready to download. This is how we implemented the idea of making a secure native application with few simple steps. Next, we perform trial and errors to make the app more user friendly for native users.