Configure the SDK

The configuration process of the SDK has severable responsibilities:

- Validation of the license key
- Validating the currently present files
- Downloading or updating the files required by the SDK to do recognition
- Preparing the files to be used for inference

SDK Configure

- **`useEffect`**: Runs the `configure` function once when the component mounts (because of the empty dependency array `[]`).
- **`PassioSDK.configure`**: Calls the Passio SDK configuration method with a key and options (`debugMode`, `autoUpdate`).
  - **If the SDK is ready** (`status.mode === 'isReadyForDetection'`), `loadingState` is updated to `'ready'`.
  - **If there's an error** (`status.mode === 'error'`), the state is set to `'error'`.
  - If the status is `'notReady'`, nothing is done.
- **Error handling**: If any error occurs during the configuration, the `catch` block sets the `loadingState` to `'error'`.

```
export type SDKStatus = 'init' | 'downloading' | 'error' | 'ready'

const [sdkStatus, setSDKStatus] = useState<SDKStatus>('init')

  useEffect(() => {
    async function configure() {
      try {
        const status = await PassioSDK.configure({
          key: "Your Passio Key",
          debugMode: false,
          autoUpdate: true,
        })
        switch (status.mode) {
          case 'notReady':
            return
          case 'isReadyForDetection':
            setSDKStatus('ready')
            return
          case 'error':
            console.error(`PassioSDK Error ${status.errorMessage}`)
            setSDKStatus('error')
            return
        }
      } catch (err) {
        console.error(`PassioSDK Error ${err}`)
        setSDKStatus('error')
      }
    }
    configure()
  }, [])
```

Track the progress of downloading Passio SDK model files

This is a method (from the Passio SDK) that registers callbacks for events related to model downloading: completedDownloadingFile: This callback is triggered when a file finishes downloading. The object passed to it contains filesLeft, which represents how many files are still left to download.

```
const [leftFile, setDownloadingLeft] = useState<number | null>(null)

  useEffect(() => {
    const callBacks = PassioSDK.onDownloadingPassioModelCallBacks({
      completedDownloadingFile: ({ filesLeft }: CompletedDownloadingFile)
        setDownloadingLeft(filesLeft)
      },
      downloadingError: ({ message }: DownloadingError) => {
        console.log('DownloadingError ===>', message)
      },
    })
    return () => callBacks.remove()
  }, [])
```

Camera authorization

You can use this hook in any functional component to check whether the app has camera authorization. For example:

```
export const useCameraAuthorization = () => {
  const [authorized, setAuthorized] = useState(false)

  useEffect(() => {
    async function getAuth() {
      const isAuthorized = await PassioSDK.requestCameraAuthorization()
      setAuthorized(isAuthorized)
    }
    getAuth()
  }, [])

  return authorized
}
```

How to handle configuration in the UI

In this component, `PassioConfigurationView`, you're conditionally rendering different UI elements based on the `sdkStatus` and `leftFile` values. Here's a breakdown of how this works:

Switch on `sdkStatus`:

- The component switches between different UI layouts based on the current value of `sdkStatus`:
    - **Case `'ready'`**: When `sdkStatus` is `"ready"`, it displays a UI element that says **SDK is Ready**.
    - **Case `'error'`**: If `sdkStatus` is `"error"`, it shows an error message **SDK is Error**.
    - **Default case**: If `sdkStatus` is anything else (including when it's `'init'` or `'downloading'`), it shows an `ActivityIndicator` (a loading spinner) and, if `leftFile` is not `null`, it displays how many files are left to download.

```
export const PassioConfigurationView = () => {

  // Rest of the Code //
  return (
    <View style={styles.container}>
      {(() => {
        switch (sdkStatus) {
          case "ready":
            return (
              <View style={styles.middle}>
                <Text>SDK is Ready</Text>
              </View>
            );

          case "error":
            return (
              <View style={styles.middle}>
                <Text>SDK is Error</Text>
              </View>
            );

          default:
            return (
              <View style={styles.middle}>
                <ActivityIndicator
                size={20}
                color={"white"}
              />
              {leftFile!==null && <Text>{`Downloading file left ${leftFil
              </View>
            );
        }
      })()}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "rgba(79, 70, 229, 1)",
  },
  middle: {
    position: "absolute",
    top: 0,
    bottom: 0,
    justifyContent: "center",
    alignItems: "center",
    alignSelf: "center",
  },
});
```

Check out the full code here. FULL CODE