iOS Lens Checks

Adding Lens Checks SDK to your project

Note: The below steps assume that you already have Cocoapods installed and initialized in your project. If you're new to Cocoapods, visit the official site to get started.

1. Open your project's *Podfile*
2. Add the Veryfi private Cocoapods repository as a source at the top of the *Podfile*

```
source 'git@bitbucket.org:veryfi/veryfi-lens-podspec.git'
```

3. Add the VeryfiLens pod to your target

```
pod 'VeryfiLens-Cheques'
```

4. A minimal version of your *Podfile* should look similar to this:

```
source 'git@bitbucket.org:veryfi/veryfi-lens-podspec.git'
source 'https://github.com/CocoaPods/Specs.git'

target 'VeryfiLensExample' do
    use_frameworks!

    # Pods for VeryfiLensExample
    pod 'VeryfiLens-Cheques'
end
```

5. Make sure your SSH key has been granted access to Veryfi's private Cocoapods repository here.

   Also make sure your SSH key has been added to ssh-agent by running this command in the Terminal:

```
# Replace /path/to/private_key with the actual path to your SSH
private key
ssh-add -K /path/to/private_key
```

6. Install the pod by running this command in the Terminal, in the root folder of your project:

```
pod install
```

Configuring your project to use Lens SDK

Add the following permissions to your app's plist:

```
<key>NSCameraUsageDescription</key>
<string>Scan documents</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>Access photo gallery for document backups</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>Access photo gallery for document uploads</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>Access photo gallery for document backups</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>Access photo gallery for document uploads</string>
<key>NSSpeechRecognitionUsageDescription</key>
<string>Quickly add transactions via voice dictation</string>
<key>NSMicrophoneUsageDescription</key>
<string>Quickly add transactions via voice dictation</string>
<key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
<string>Helps to identify places around you</string>
<key>NSLocationWhenInUseUsageDescription</key>
<string>Helps to identify places around you</string>
<key>NSContactsUsageDescription</key>
<string>Add your ___@veryfi.cc assigned email address for reference<
/string>
<key>NSCalendarsUsageDescription</key>
<string>Enrich your data with business meetings and events from your
Calendar</string>
```

Initializing Lens

1. Import required symbols from Lens SDK:

```
import VeryfiLens
```

2. Configure your authentication credentials:

```
let CLIENT_ID = "XXX" // replace XXX with your assigned Client Id
let AUTH_USERNAME = "XXX" // replace XXX with your assigned
Username
let AUTH_APIKEY = "XXX" // replace XXX with your assigned API Key
let URL = "XXX" // replace XXX with your assigned Endpoint URL

let veryfiLensCredentials = VeryfiLensCredentials(clientId:
CLIENT_ID,
                                                  username:
AUTH_USERNAME,
                                                  apiKey:
AUTH_APIKEY,
                                                  url: URL)
```

3. Mandatory Lens Checks SDK settings. More available settings later in this section.

```
let veryfiLensSettings = VeryfiLensSettings()
veryfiLensSettings.autoRotateIsOn = true
veryfiLensSettings.documentTypes = ["check"]
veryfiLensSettings.showDocumentTypes = true
veryfiLensSettings.moreMenuIsOn = false
```

4. Enable background upload support. Add the below to your AppDelegate:

```
import AWSS3

func application(_ application: UIApplication,
handleEventsForBackgroundURLSession identifier: String,
completionHandler: @escaping () -> Void) {
    // Store the completion handler
    AWSS3TransferUtility.interceptApplication(application,
handleEventsForBackgroundURLSession: identifier,
completionHandler: completionHandler)
}
```

5. Initialize Lens:

```
VeryfiLens.shared().configure(with: veryfiLensCredentials,
settings: veryfiLensSettings)
```

Available settings:

`docDetectFillUIColor`: document detection rectangle fill color (default: *"#9653BF8A"*)

`docDetectStrokeUIColor`: document detection rectangle stroke color (default: *null*)

`locationServicesIsOn`: enables/disables location services to grab user's lat & lng (default: *true*)

`galleryIsOn`: enables/disables the gallery in the camera view (default: *true*)

Launching Lens

Launch the Lens camera:

```
1. VeryfiLens.shared().showCamera(in: self)
```

Your app will need to communicate with Lens to handle user actions, various status changes and extraction results from Veryfi. See the Communicating with Lens section below for details.

Communicating with Lens

1. Set your delegate:

```
VeryfiLens.shared().delegate = self
```

2. Implement the delegate methods:

```
extension ViewController: VeryfiLensDelegate {
    func veryfiLensClose(_ json: [String : Any]!) {
        let jsonData = try? JSONSerialization.data(withJSONObject:
json as Any, options: .prettyPrinted)
        let jsonString = String(data: jsonData!, encoding: .utf8)
        print(String("veryfiLensClose: " + jsonString!))   // do
something with the JSON here
    }

    func veryfiLensUpdate(_ json: [String : Any]!) {
        let jsonData = try? JSONSerialization.data(withJSONObject:
json as Any, options: .prettyPrinted)
        let jsonString = String(data: jsonData!, encoding: .utf8)
        print(String("veryfiLensUpdate: " + jsonString!))   // do
something with the JSON here
    }

    func veryfiLensSuccess(_ json: [String : Any]!) {
        let jsonData = try? JSONSerialization.data(withJSONObject:
json as Any, options: .prettyPrinted)
        let jsonString = String(data: jsonData!, encoding: .utf8)
        print(String("veryfiLensSuccess: " + jsonString!))   // do
something with the JSON here
    }

    func veryfiLensError(_ json: [String : Any]!) {
        let jsonData = try? JSONSerialization.data(withJSONObject:
json as Any, options: .prettyPrinted)
        let jsonString = String(data: jsonData!, encoding: .utf8)
        print(String("veryfiLensError: " + jsonString!))   // do
something with the error JSON here
    }
}
```

Delegate Definitions

- `veryfiLensClose` - the Veryfi Lens camera has been closed, either as a result of submitting an image for processing, or the user closed the camera without submitting an image.

  Sample data:

```
{
    "status": "close",
    "queue_count": 1,
    "framework-version": "1.5.7",
    "session_scan_count": 1,
    "framework-build": "49"
}
```

NOTE: In the object above, `queue_count` refers to the number of submitted documents that are currently in the processing queue. `session_scan_count` refers to the number of documents that were submitted in the most recent Lens camera session - if this is equal to 0 (zero) then the camera session was canceled without submitting anything.

- `veryfiLensUpdate` - during the processing of a document, this delegate will be fired multiple times. One time it will contain the thumbnail path for the submitted document and one time it will contain a full-size image path. In addition, it would let you know when the package was removed.

Sample *package created* notification:

```
{
    "status": "start",
    "package_id": "5kY9hXlb0COeg5n07y71"
}
```

Sample *thumbnail path* notification:

```
{
    "status": "inprogress",
    "msg": "img_thumbnail_path",
    "data": "/path/to/thumbnail.jpg",
    "package_id": "5kY9hXlb0COeg5n07y71"
}
```

Sample *full-size image path* data:

```
{
    "status": "inprogress",
    "msg": "img_original_path",
    "data": "/path/to/image.jpg",
    "package_id": "5kY9hXlb0COeg5n07y71",
    "document_type": "check"
}
```

Sample *upload progress percentage* data:

```
{
    "status": "inprogress",
    "msg": "progress",
    "data": 68,
    "package_id": "5kY9hXlb0COeg5n07y71"
}
```


Sample *package removed* notification data:


```
{
    "status": "removed",
    "msg": "clear_package",
    "package_id": "5kY9hXlb0COeg5n07y71"
}
```


- `veryfiLensError` - if an error occurs during uploading or processing a submitted or a general exception or crash is caught in Veryfi Lens, this notification contains the error details.

  Sample *error* data:


```
{
    "status": "error",
    "package_id": "5kY9hXlb0COeg5n07y71",
    "msg": "Error description"
}
```


- `veryfiLensSuccess` - this delegate fires once a document has finished processing, whether it was submitted via the camera, the gallery. This delegate provides the response from the Veryfi API.

  Sample data:

```
{
  "data" : {
    "amount_text" : "Seven hundred fighteen and is",
    "payer_address" : "765 Dolor sit Amet APT B5\nBrooklyn, NY,
12345",
    "pdf_url" : "https:\/\/scdn.veryfi.com\/checks\/2be35587-c1ac-
4e9b-95a4-fb3ee25572f4\/9ce691d4-a31a-4552-b17f-4445e8b41b47.pdf?
Expires=1647392082&Signature=Whyq6BIaMg1QweK8daAFRh3K558mrvV63tAAxp
mPCOeEogFFYOysEGkeVjXnkUH8MiqxtjEfaVvY6f-
ypYcbr6OngFh~1kPkL0n5NJ8vD3bDHbjnTBw9QqVLiRISiOQ1xaPoLkRdiB6mHZLS-
1Xhb5yj8UqG1jDjy6KQnI3Z2B4p8f7v51Kf979XjLRBoWAkEYwVv5FF4fcm-
Koa6oHfK-wkPLBKzSIizb3TmLMGiIioebnFQ5lQ0CWmPasUejMF43V~r3eMtE7vjy-
akB6RdfPlM9ezxPuFkI6~7LgTUHK9BJl962jR5gscaQCr32zEJ2Ud1ZkpJxJ3qu8dJk
taRA__&Key-Pair-Id=APKAJCILBXEJFZF4DCHQ",
    "receiver_name" : "Mary Johnson",
    "bank_address" : null,
    "bank_name" : null,
    "fractional_routing_number" : null,
    "check_number" : "0007",
    "check_account_routing" : "C950780010C
A654301008A80008765930100C",
    "date" : "2019-08-11",
    "receiver_address" : null,
    "payer_name" : "John Smith",
    "amount" : 715.4,
    "memo" : "Monthly rent"
  },
  "document_type" : "check",
  "status" : "done",
  "package_id" : "5kY9hXlb0COeg5n07y71"
}
```


Prepare your app for the App Store

1. Go to your app's *Target* > *Build Phases*

   Make sure VeryfiLens.xcframework is included in the *Embedded Frameworks* section

Please note: adding the iOS Lens Checks SDK to your app will increase your final app size by up to 17 MB. This is due to machine learning models, support libraries, etc included in the SDK.