How to let users send feedback using email app of their choice in iOS

#blog

If you have an app on the App Store chances are you also have option in settings possibly to let users send you feedback via email. The default is to show system view controller which works as an extension for Apple Mail.
That may be fine but many users are using Gmail, Outlook or something else which they would rather use. I am going to show you, how to let user pick their email app of choice and fill the message with basic info.

Creating system URLs

It involves a few steps. First we will create system URLs to open other apps based on the known schemas they are using. For Gmail, Microsoft Outlook and Readdle Spark this will look like this:

```
let to = “your@email.com”
let subjectEncoded =
subject.addingPercentEncoding(withAllowedCharacters: .urlHostAllowed)!
let bodyEncoded =
body.addingPercentEncoding(withAllowedCharacters: .urlHostAllowed)!
let gmailUrl = (URL(string: “googlegmail://co?to=\(to)&subject=\
(subjectEncoded)&body=\(bodyEncoded)”), “Gmail”)
let outlookUrl = (URL(string: “ms-outlook://compose?to=\(to)&subject=\
(subjectEncoded)&body=\(bodyEncoded)”), “Outlook”)
let sparkUrl = (URL(string: “readdle-spark://compose?recipient=\(to)&subject=\
(subjectEncoded)&body=\(bodyEncoded)”), “Spark”)
```

I am using tuples so the example is shorter. They each contain system URL for the email app and friendly name which will be displayed to the user.
You can google other schemas using the keywords: “[app name] ios url schema”.
There is no point in offering Gmail choice if the user does not have Gmail app installed. So prior showing the option, we will filter out system URLs to keep only those that can be opened by iOS.
That can look like this:

```
let availableUrls = [gmailUrl, outlookUrl, sparkUrl].filter { (item) -> Bool in
   return item.0 != nil && UIApplication.shared.canOpenURL(item.0!)
}
```

Just to be safe we check for `nil` and then use system method `canOpenURL` to determine if the URL can be opened.

Since those we know these URLs are no longer optional, we can use something like this:

```
return availableUrls as! [(url: URL, friendlyName: String)]
```

To get tuple with non-optional URLs assuming we have all this code encapsulated in a method.

Assuming we have at least one available URL to open, we can let user pick with `UIAlertController` like this:

```
func showEmailOptions(for items: [(url: URL, friendlyName: String)]) {
        let ac = UIAlertController(title: nil, message: nil,
preferredStyle: .actionSheet)
        ac.addAction(UIAlertAction(title: "Apple Mail", style: .default,
handler: { (_) in
            self.openMailApp()
        }))

        for item in items {
            ac.addAction(UIAlertAction(title: item.friendlyName,
style: .default, handler: { (_) in
                UIApplication.shared.open(item.url, options: [:],
completionHandler: nil)
            }))
        }
        ac.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler:
nil))

        present(ac, animated: true)
}
```

The first option is Apple Mail which does not use URL but instead has custom `MFMailComposeViewController` to handle sending email.

Sending email with Apple Mail

Here is quick example usage:

```
 func openMailApp() {
        let composer = MFMailComposeViewController()

        if MFMailComposeViewController.canSendMail() {
            composer.mailComposeDelegate = self
            composer.setToRecipients([“your@email.com”])
            composer.setSubject(“Your great app support”)
            composer.setMessageBody(“Hello, “, isHTML: false)

            present(composer, animated: true, completion: nil)
        }
}
```

You need to conform to its delegate `MFMailComposeViewControllerDelegate` and implement `didFinishWith` method like so:

```
func mailComposeController(_ controller: MFMailComposeViewController,
didFinishWith result: MFMailComposeResult, error: Error?) {
        controller.dismiss(animated: true)
    }
```

The `dismiss` call is required otherwise users cannot leave the mail controller. You can also check the `result` and maybe thank user for sending feedback.

And that is all!

Some tips

I recommend adding at least app version to the support email so you have some sense what the user is currently using. You can get if from the `Bundle` like this:

```
Bundle.main.object(forInfoDictionaryKey: “CFBundleShortVersionString") as?
String
```

I would highly recommend getting additional diagnostic info about the device like the model or iOS version, language etc and pre-fill this to the email body. Plus explain to your user what this is and why you need it to better help them.

Thanks for reading!