ECS 189E Homework 3

Overview

In the homework 3, we are fixing OTP bug, simplifying the login for our "old users" and building up the home view that presents the wallet of the user.

For this assignment, your app's OTP text field should be able to handle all kinds of partial deletion and partial entering.

For the very last successfully logged in user on this device, the App should offer them a more convenient way to login: pre-fill the phone number and skip the login (sms verification) in this case.

On the wallet view, after user successfully logged in, the App should present a list of information of the wallet, and also allow the user to change the user name on the view.

Usage of Provided Code

No matter you are using your own code or the starter kit, you will need those three files in the "Support" folder. **The Api.swift is also updated, so remember to replace the old one.** Make sure you copy the folder to your project folder and also add the folder to the project using Xcode. You do not need to edit any file that is in the "Support" folder.

classes.swift

This file contains all the customized classes that you will use for this homework. It is very important to understand these classes first.

Storage

This struct helps you to store two variables locally on your device (or simulator). You can use this two variables as all the other normal variables that you defined. However, this struct redefined the `set` and `get` of these two variables. As a result, whenever you set the variables or get value from them, additional steps will be made. And the value that is passed to the variables will be saved to device, or when reading from the variables, the value will be read from device. The usage is provided as comments in the file.

Wallet and Account

As you can see, this class defined a wallet that contains all the information related to the logged in user. It contains a user name, a phone number, a total amount, and a list of accounts. For each account, there is a unique ID, a name and an amount.

A function that you definitely will be using is the second `init` of `Wallet`. By passing the `response`, which will be explained in Api.swift section, this `init` will parse the data that is received from Api call and setup a wallet object for you. With passing `true` to the `ifGenerateAccounts`, the `init` will generate a random number of accounts for you, each contains a random amount, and the total amount of the wallet will be updated according to it. **The list of accounts will change each time you logged in since they are not stored on server. However, other information should remain the same.** If you want to customized your accounts, pass `false` to the `ifGenerateAccounts`, other information will still be parsed for you while for the accounts, you will have an empty array. The usage is provided as comments in the file.

Api.swift

Response

After you used `Api.verifyCode` and successfully verified the user without any error, the `response` contains the information that related to the phone number that is sent back from the server.

```
["status": ok, "auth_token": oKhoMfhkCpxmtVT_Jt1TnCXv2pxPtITD, "user": {
    "drivers_license" = "<null>";
    "e164_phone_number" = "+15306018718";
    email = "<null>";
    "keyfile_password" = "BtEik_-jGpZEwHPZD4rV1YIm5XuXB5Zh";
    name = op;
    "payment_methods" =     (
    );
    "tos_accpted" = 0;
    "user_id" = ahJzfmVjczE4OWUtZmFsbDIwMThyEQsSBFVzZXIYgICAgLyhggoM;
}, "is_new_user": 0]
```

Auth token

Attention. Attention. Attention. A unique auth token is signed to a user after successfully logged in. On the server, it is the key to verify and get all the user information. You can get the auth token from response using `let authToken = response["auth_token"] as? String`.

If you read through the file, you will see that we used a different way to set the `URLSession`. Now we are using a configuration. The most important part of the configuration is the `httpAdditionalHeaders`. This is where we pass the authToken **if it is stored in the device.**

New Functions

If an auth token can be found on the device, `Api.user()` will get you the user information related to this token with the `response`. And `Api.setName(name: inputString)` will update the user name related to this token on the server.

Extension.swift

It contains a subclass `PinTextField` of `UITextField` and a protocol `PinTextFieldDelegate` which will indicate whenever the backspace is tapped.

Homework Starts Here

Fix OTP Text Field

Change the class of all of your 6 text field to `PinTextField` and implement the `PinTextFieldDelegate` protocol to fix the deletion bug. **Your app is required to handle all kinds of partial deletion and partial entering properly in this assignment**.

Preparation

After successfully verified the sms code, store the phone number in E164 and auth token using `Storage`, in order to make sure that you can get the user information and update the user name by Api call in the future steps.

Wallet View

Use `Api.user()` to get the user information from server on this view, and initialize a `Wallet` object using the provided `init` function and the response.

You need to at least present:

1. The user name,
2. Total amount
3. The list of accounts on this view. For each account, **you need to at least present: the name and amount of the account**. **It is required to use a table view or collection view to present the list of accounts.** No user interaction is required for the table view or collection view. It is for presentation only.

Hint: You only need the `UITableViewDataSource` or `UICollectionViewDataSource`.

Update The User Name

Your app is required to let the user be able to change the user name in his or her wallet, and update the user name that is on the server using the Api function provided. As a result, when user log back in again, the wallet view should present the new user name.

Present user's phone number in e164 format as the default name if the user didn't set a name or if the user set the name as empty string.

Simplify Login Process

Pre-fill the phone number

You app is required to pre-fill the last logged in user's phone number on the login view. The pre-filled phone number should still be a formatted phone number.

Hint: use `Storage.phoneNumberInE164` and string manipulation.

Skip verification

On the login view (first view), test if current user is the last successfully logged in user, and **your app is required to skip the verification process for the user**. By skipping the process, your App should take the user to the wallet view directly if the user is the last successfully logged in user.

Hint: use the following code

```
if Storage.authToken != nil, Storage.phoneNumberInE164 ==
EnteredPhoneNumberInE164 {
  // This user is the last successfully logged in user.
}
```

Navigation

It is very important to meet the user's expectation when transfer among the views. You want to make sure the user will not go to Verify View when coming back from Wallet View.

1. New User: Login View -(valid number)-> Verify View -(valid code)-> Wallet View -(logout)-> Login View
2. Old User: Login View -(valid last user)-> Wallet View -(logout)-> Login View

Other Requirements

You can only use `Storage` to store any data to the device. In another word, it is not allowed to save any user information locally besides the phone number and the auth token.

Coding Style

It is always very important to make sure that your code is easy to read and understand. The following points will be considered when your coding style is graded:

1. Reasonable names for ViewControllers, views and objects.
2. Reasonable names for files, functions and variables.
3. Necessary comments.
4. Reasonable order of the functions.
5. The use of optional (Question mark and exclamation mark).

Documentation

Write a README.md to briefly state what functions does the App have and how did you implement them. Imagine yourself working in a big company, and this document is for the people who will take over your work, or will become your partner and work on this App with you. Remember to include your name and student ID in the document.

Grading

1. UI Design and Auto Layout (5')
2. OTP text field functionality (10')
3. Wallet View and Table View (15')
4. Update User Name (5')
5. Simplify Login Process (10')
6. Style and Documentation (5')

Submission

Push your files to the repository. Submit the url for the respository to Canvas.