Modularising a monolith!

Araks Avoyan
iOS Engineer @ Spotify

My team owns authentication technical stack on iOS (which I'll refer to as the Login feature). In short we are helping iOS and Android users to login into Spotify app.

Spotify app launched in 2012

LoginStateController was the M(assive)VC that was part of Login feature starting from 2012.

55424 lines of Obj - C

```
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
➜  login git:(63e45ba9a985) ✗
```

We wanted to update our tech stack and migrate to Swift

Legacy code

Legacy code, broken behaviour

Legacy code, broken behaviour and undesired strong coupling

PROBLEM

Dealing with legacy code

PROBLEM

Ownership alignment

GOALS

GOALS

- Identify and separate core responsibilities
- Design clear interfaces
- Improve existing internal public interface
- Enable gradual migration

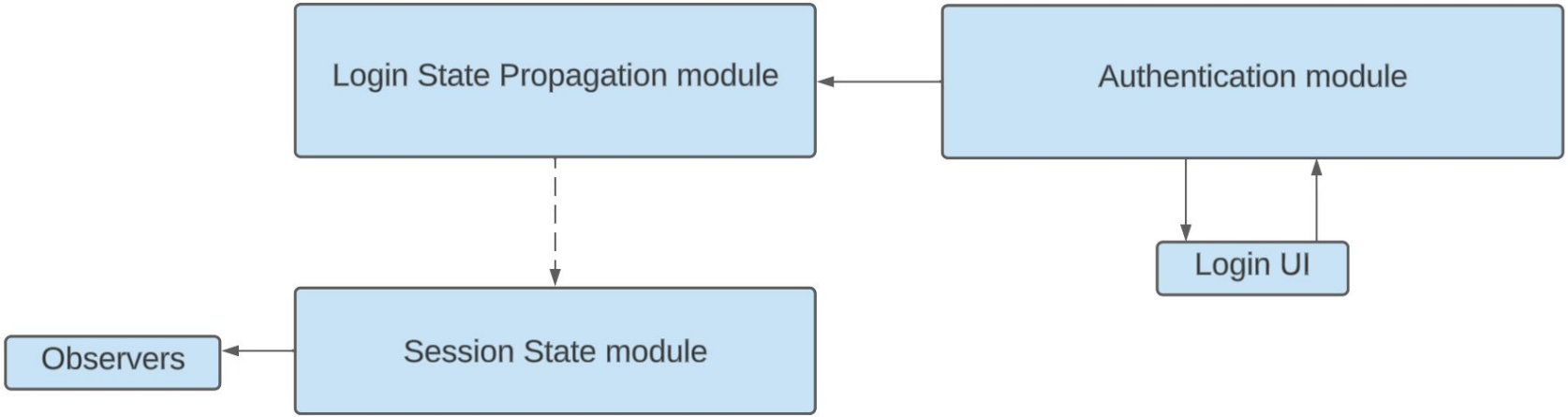
MAIN RESPONSIBILITIES

- Authentication responsibility**:** provide a way for the user to authenticate (or login).
- Session state responsibility: configure application based on the user being authenticated or not.
    i. Switch app from logged in to logout state
    ii. Providing the HTTP client
    iii. Notifying consumers of session state changes

Responsibilities were all intertwined

The resulting architecture was formed around 3 key modules –

The resulting architecture was formed around 3 key modules – **Authentication**, **Login State Propagation** and **Session State**.



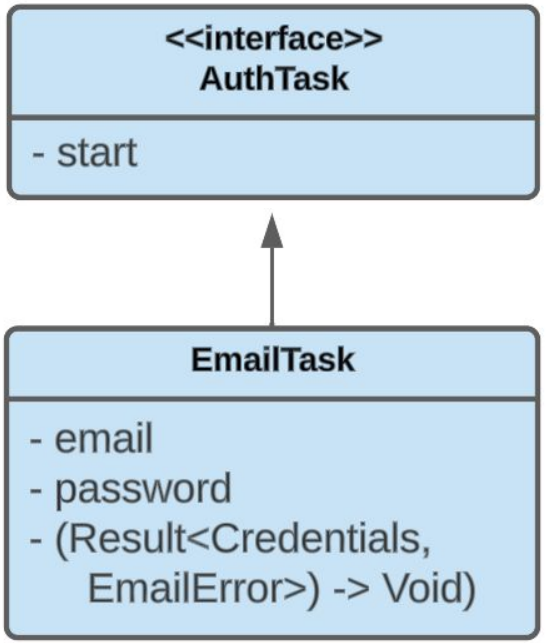
Authentication

The authentication module is responsible for running authentication requests

Authentication module: Task

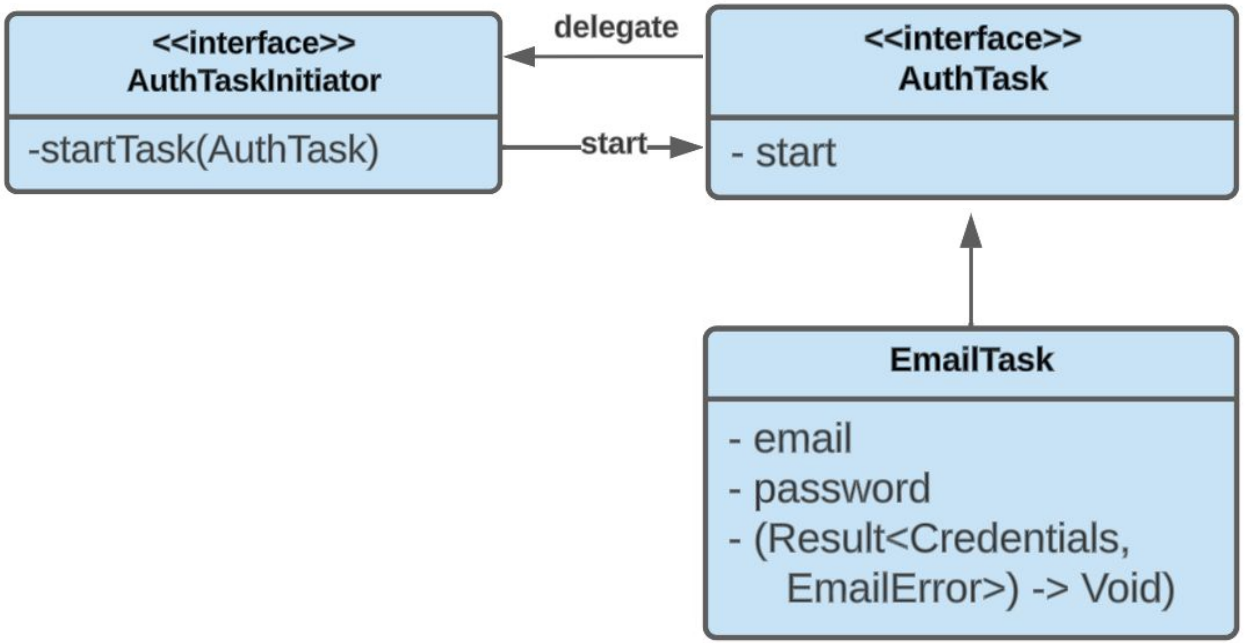
Authentication module: Task

- Data
- Callback
- Custom logic



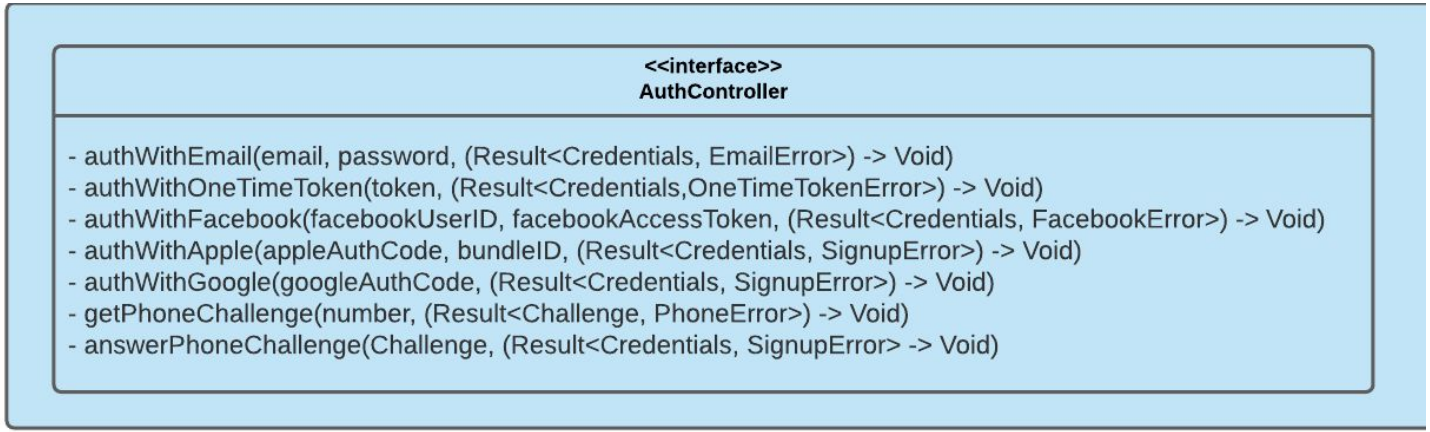
Authentication module: Task

- Data
- Callback
- Custom logic



Authentication module: Alternative solution

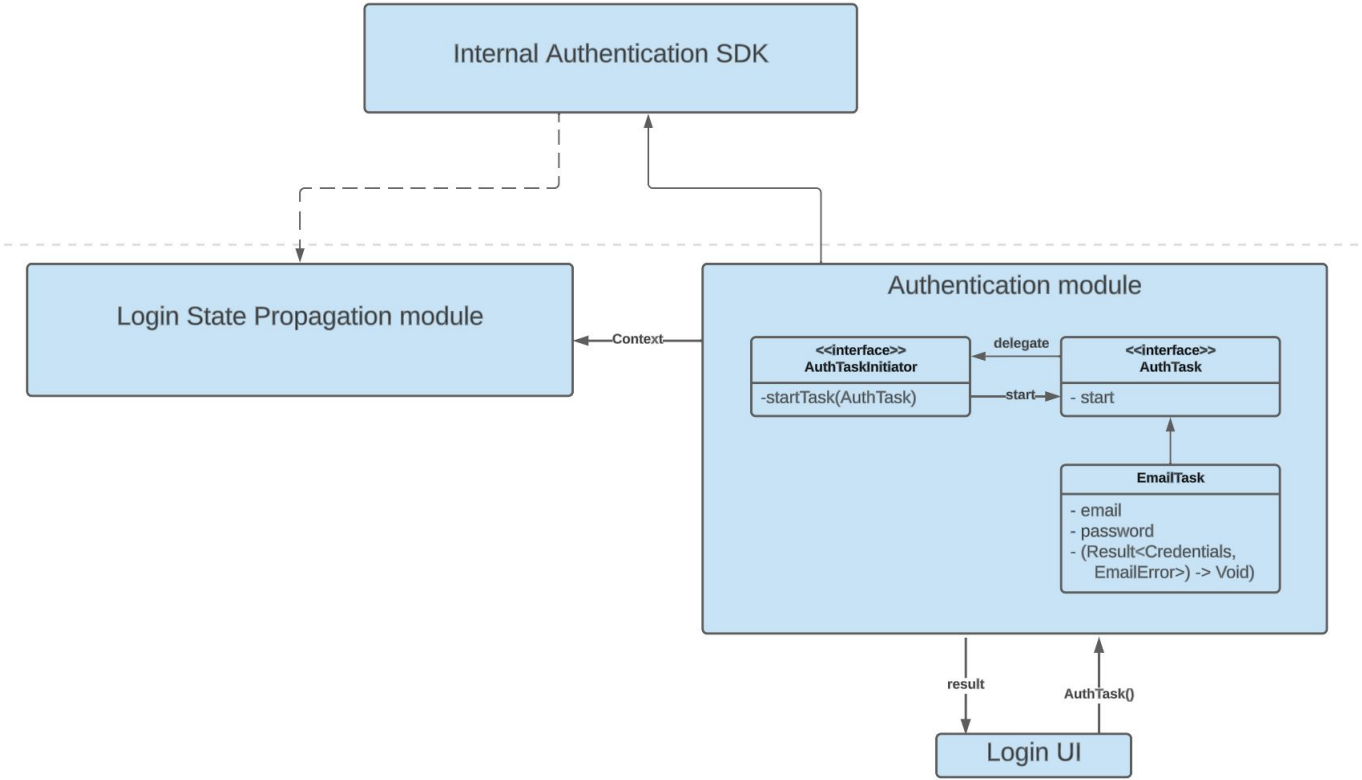
Instead of creating a new type for each of the login tasks, they can be made available through a single API.



Background information

Internal Authentication SDK

Authentication module: Communication

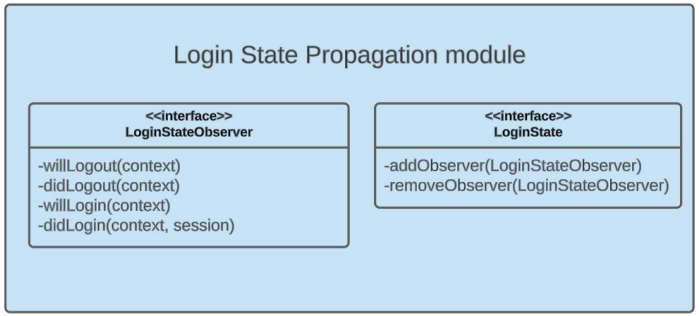


Login State Propagation

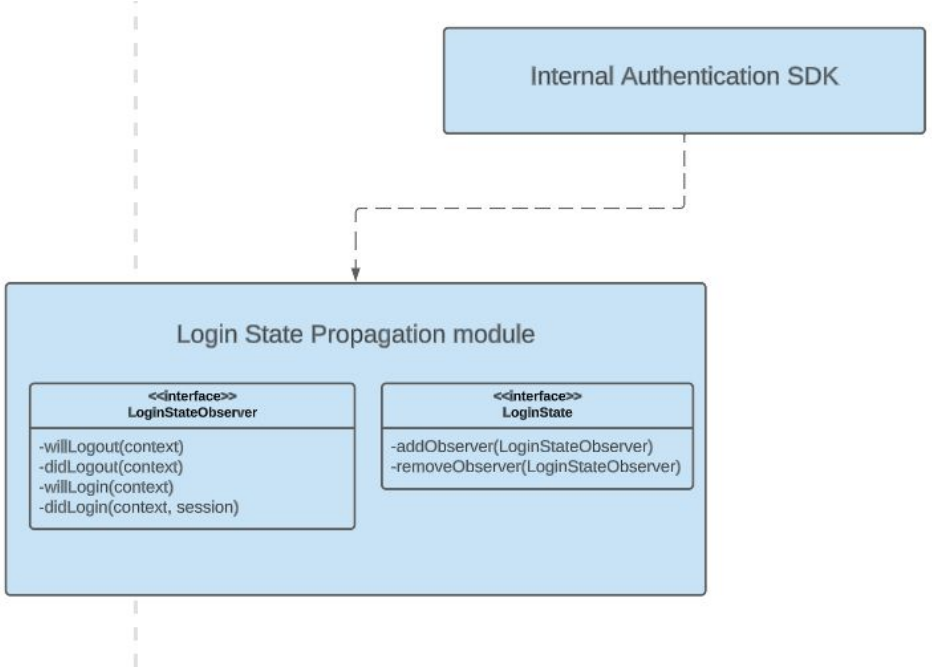
Internal Auth SDK :

- is about to start authentication
- did authenticate
- is about to start log out
- did finish logout

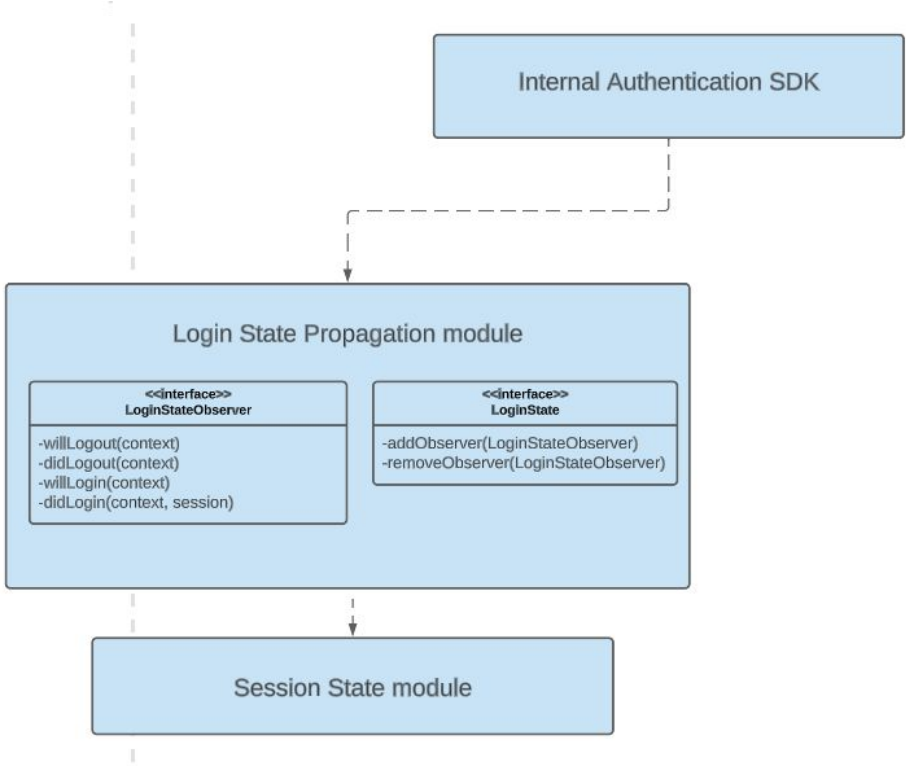
Login State Propagation Module: Communication



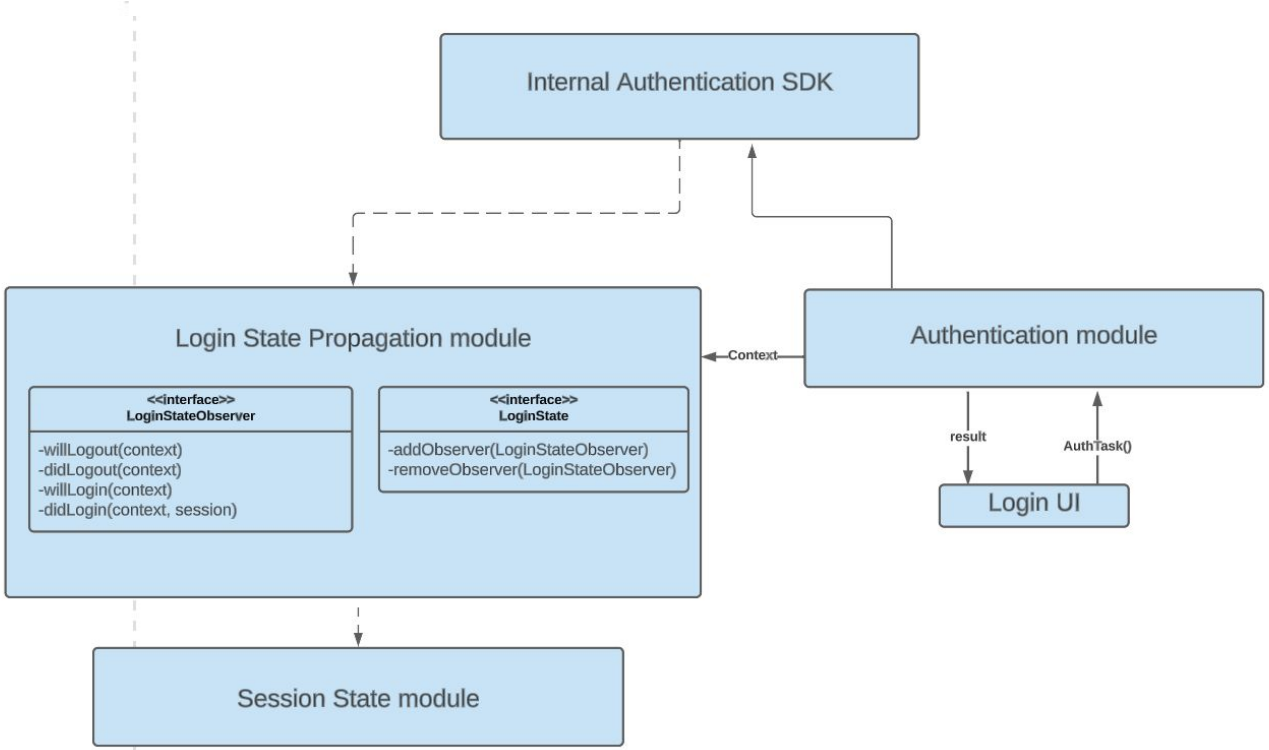
Login State Propagation Module: Communication



Login State Propagation Module: Communication



Login State Propagation Module: Communication



Session State Module

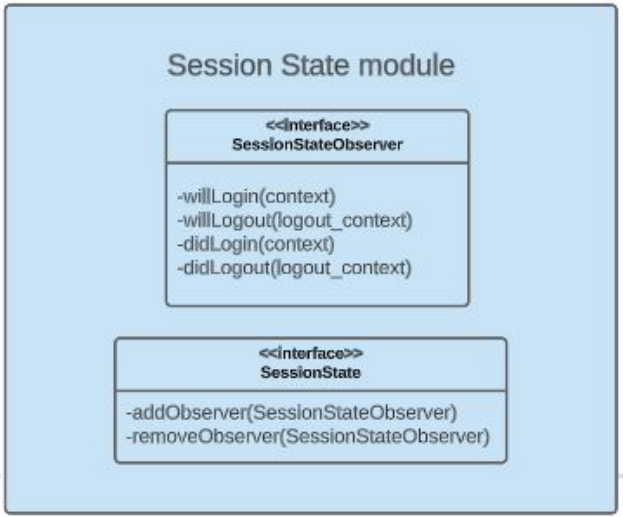
Session State Module: Responsibilities

- Observe and react to state change in **Login state propagation module**
- Switch UI between logged-in to logged-out modes
- Configure session before entering the logged-in mode
- Destroy session on exiting logged-in mode
- Notify consumers of API about change

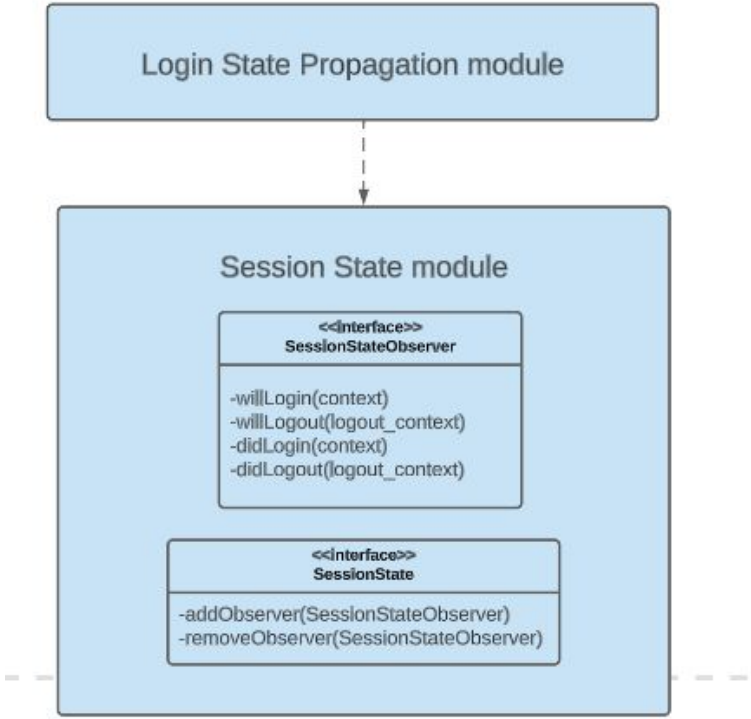
Session State Module: Why we need Context

- autologin
- manual login
- logout triggered by the user

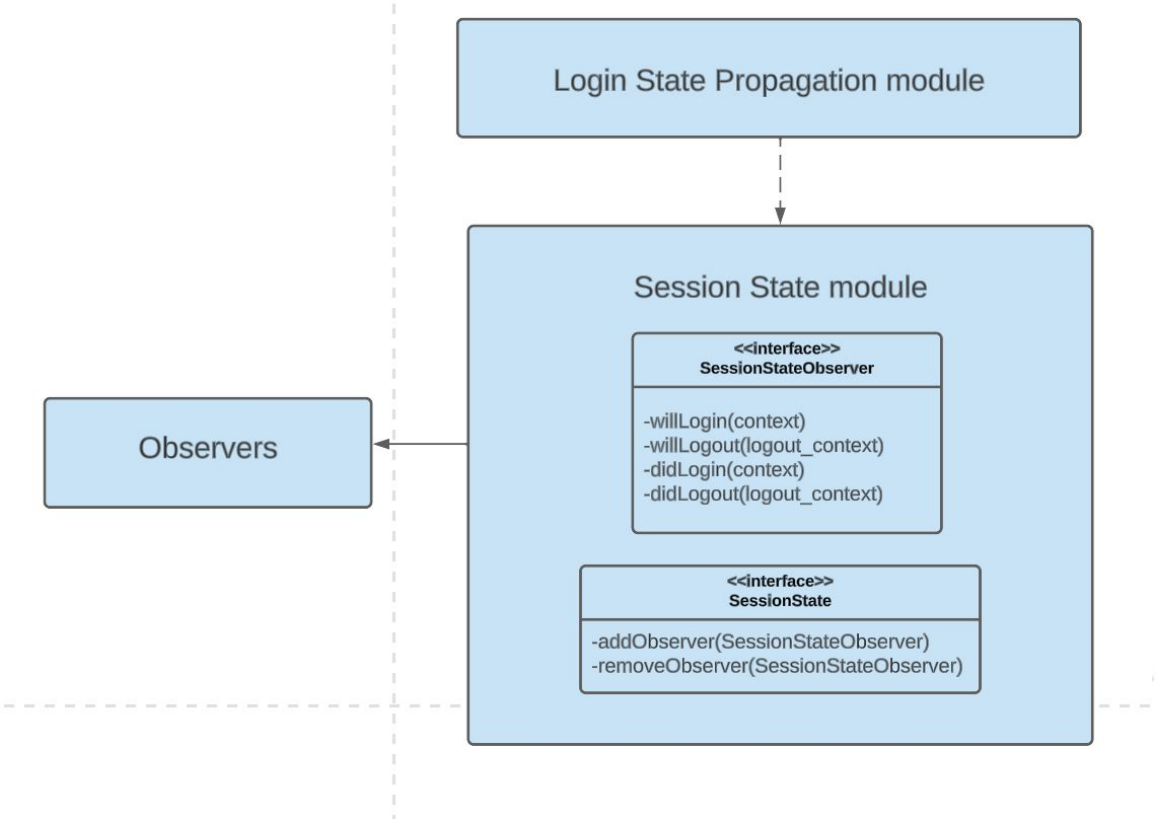
Session State Module: Communication

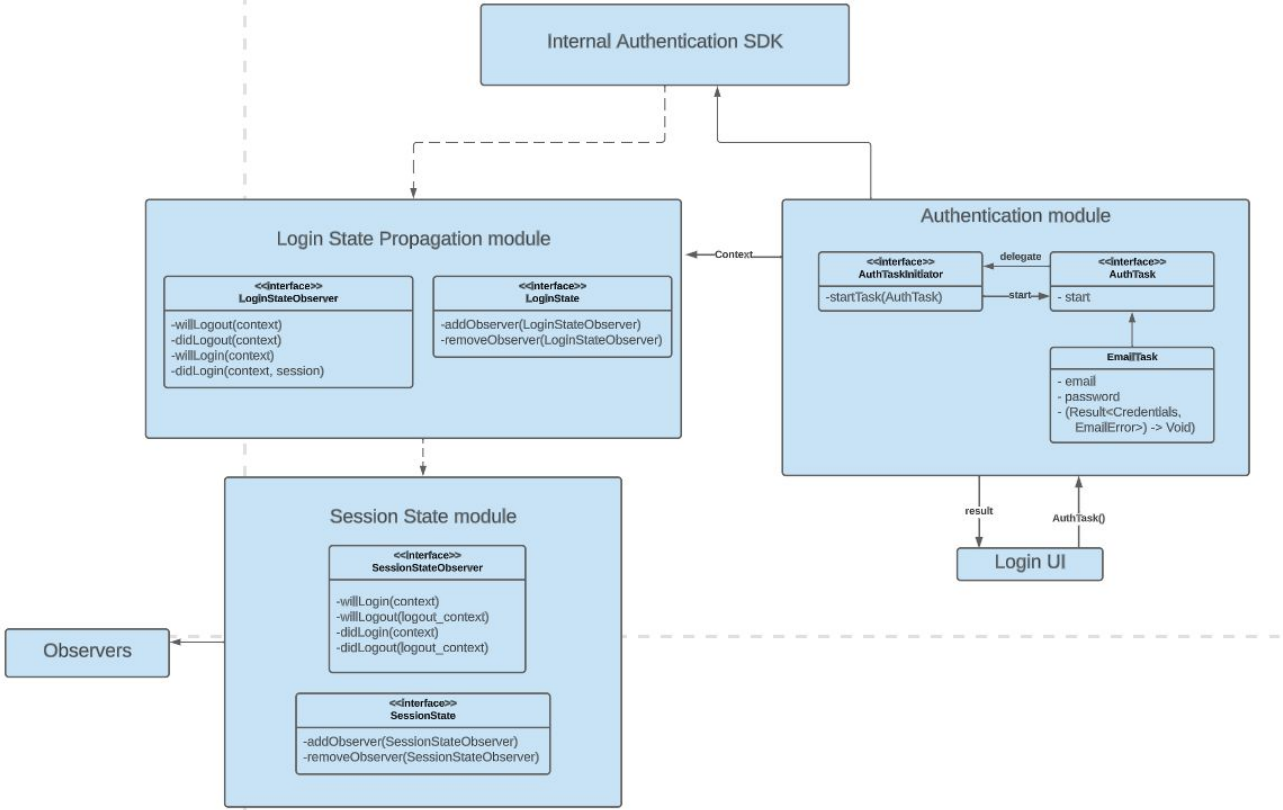


Session State Module: Communication



Session State Module: Communication




Internal Authentication SDK
Login State Propagation module
<<interface>>
LoginStateObserver
-willLogout(context)
-didLogout(context)
-willLogin(context)
-didLogin(context, session)
<<interface>>
LoginState
-addObserver(LoginStateObserver)
-removeObserver(LoginStateObserver)
Context
Authentication module
<<interface>>
AuthTaskInitiator
-startTask(AuthTask)
delegate
start
<<interface>>
AuthTask
- start
EmailTask
- email
- password
- (Result<Credentials, EmailError>) -> Void)
result
AuthTask()
Login UI
Session State module
<<interface>>
SessionStateObserver
-willLogin(context)
-willLogout(logout_context)
-didLogin(context)
-didLogout(logout_context)
<<interface>>
SessionState
-addObserver(SessionStateObserver)
-removeObserver(SessionStateObserver)
Observers

Lessons learned and things that helped us

- Brainstorming and drawing the diagrams
- A/B testing each module before the release
- No major incidents

The end result

- Manageable independent modules
- Improved our internal public interfaces
- Flexibility for adding new authentication methods
- Allowed us to migrate to new Internal Authentication SDK
- Migrated to Swift

41525 lines of Obj-C

21045 lines of Swift

Thank you!

Twitter: @Araks Avoyan