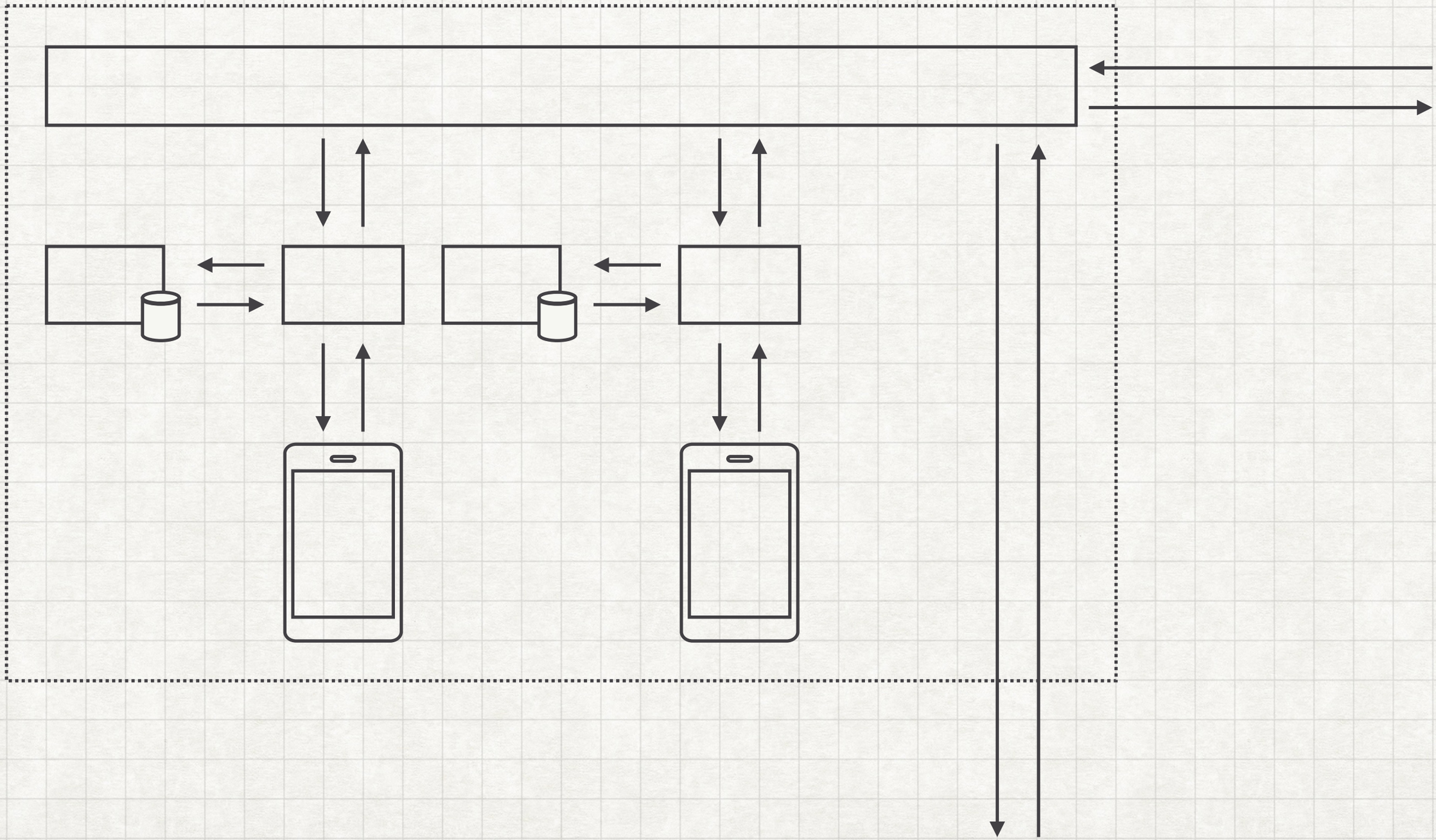
COORDINATORS

COMMENT SOLD'S MOBILE APP ARCHITECTURE



“

#PROBLEM:
MVC HAS ISSUES

”

MVC = "MASSIVE VIEW CONTROLLER"

MODEL-VIEW BINDING

SUBVIEW ALLOCATION

LAYOUT

DATA TRANSFORMATION

USER INPUT

MODEL MUTATION

NAVIGATION FLOW

DATA FETCHING

Credit: Soroush Khanlou

MVC = "MASSIVE VIEW CONTROLLER"

MODEL-VIEW BINDING

SUBVIEW ALLOCATION

LAYOUT

DATA TRANSFORMATION

USER INPUT

MODEL MUTATION

NAVIGATION FLOW

DATA FETCHING

Credit: Soroush Khanlou

```
override func collectionView(_ collectionView: UICollectionView,
    cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCell(
        withReuseIdentifier: reuseIdentifier, for: indexPath)

    guard let cell = cell as? ProductCell else { return cell }

    let product = products[indexPath.row]

    let productFormatter = NumberFormatter()
    productFormatter.numberStyle = .currency
    let priceText = productFormatter.string(from: product.price)

    cell.priceLabel.text = priceText
    cell.descriptionLabel.text = product.description
    cell.buyButton.addTarget(self, action: #selector(buyTapped),
                             for: .touchUpInside)

    return cell
}

@objc func buyTapped() {
    // add product to cart
    // go to cart page
}
```

```
override func collectionView(_ collectionView: UICollectionView,
    cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCell(
        withReuseIdentifier: reuseIdentifier, for: indexPath)

    guard let cell = cell as? ProductCell else { return cell }

    let product = products[indexPath.row]

    let productFormatter = NumberFormatter()
    productFormatter.numberStyle = .currency
    let priceText = productFormatter.string(from: product.price)

    cell.priceLabel.text = priceText
    cell.descriptionLabel.text = product.description
    cell.buyButton.addTarget(self, action: #selector(buyTapped),
                             for: .touchUpInside)

    return cell
}

@objc func buyTapped() {
    // add product to cart
    // go to cart page
}
```

```
override func collectionView(_ collectionView: UICollectionView,
    cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCell(
        withReuseIdentifier: reuseIdentifier, for: indexPath)

    guard let cell = cell as? ProductCell else { return cell }

    let product = products[indexPath.row]

    let productFormatter = NumberFormatter()
    productFormatter.numberStyle = .currency
    let priceText = productFormatter.string(from: product.price)

    cell.priceLabel.text = priceText
    cell.descriptionLabel.text = product.description
    cell.buyButton.addTarget(self, action: #selector(buyTapped),
                             for: .touchUpInside)

    return cell
}

@objc func buyTapped() {
    // add product to cart
    // go to cart page
}
```

```
override func collectionView(_ collectionView: UICollectionView,
    cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCell(
        withReuseIdentifier: reuseIdentifier, for: indexPath)

    guard let cell = cell as? ProductCell else { return cell }

    let product = products[indexPath.row]

    let productFormatter = NumberFormatter()
    productFormatter.numberStyle = .currency
    let priceText = productFormatter.string(from: product.price)

    cell.priceLabel.text = priceText
    cell.descriptionLabel.text = product.description
    cell.buyButton.addTarget(self, action: #selector(buyTapped),
                             for: .touchUpInside)

    return cell
}

@objc func buyTapped() {
    // add product to cart
    // go to cart page
}
```

```
override func collectionView(_ collectionView: UICollectionView,
    cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCell(
        withReuseIdentifier: reuseIdentifier, for: indexPath)

    guard let cell = cell as? ProductCell else { return cell }

    let product = products[indexPath.row]

    let productFormatter = NumberFormatter()
    productFormatter.numberStyle = .currency
    let priceText = productFormatter.string(from: product.price)

    cell.priceLabel.text = priceText
    cell.descriptionLabel.text = product.description
    cell.buyButton.addTarget(self, action: #selector(buyTapped),
                             for: .touchUpInside)

    return cell
}

@objc func buyTapped() {
    // add product to cart
    // go to cart page
}
```

“

SPLIT UP THE ROLES (MVVM, MVP, VIPER, ...)

”

EXAMPLE: MODEL VIEW VIEW-MODEL (MVVM)

VIEW CONTROLLER

MODEL-VIEW BINDING

SUBVIEW ALLOCATION

LAYOUT

NAVIGATION FLOW

DATA FETCHING

USER INPUT

MODEL MUTATION

VIEW-MODEL

DATA TRANSFORMATION

“

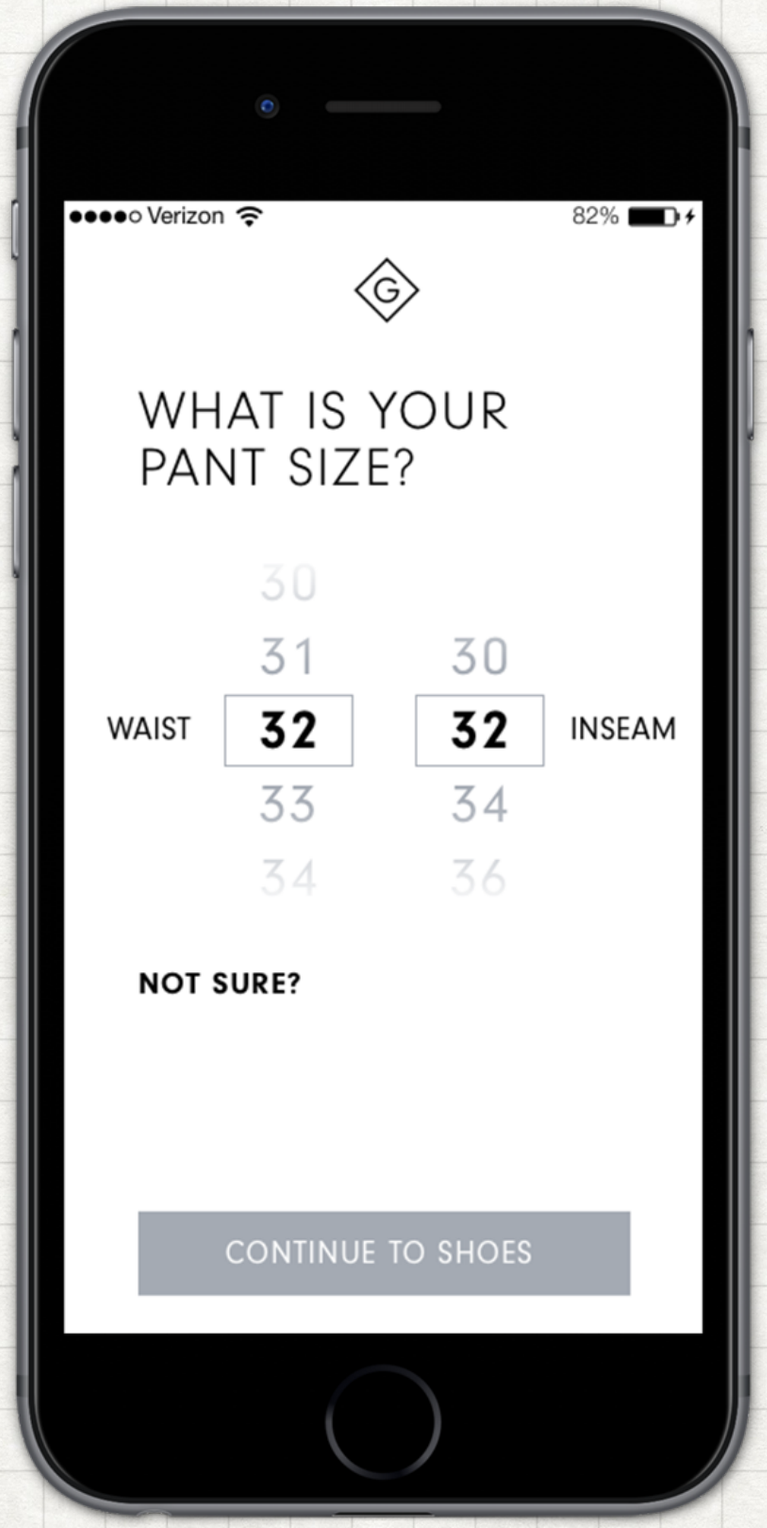
#GOAL: ALL COMPONENTS SHOULD HAVE CLEARLY DEFINED ROLES

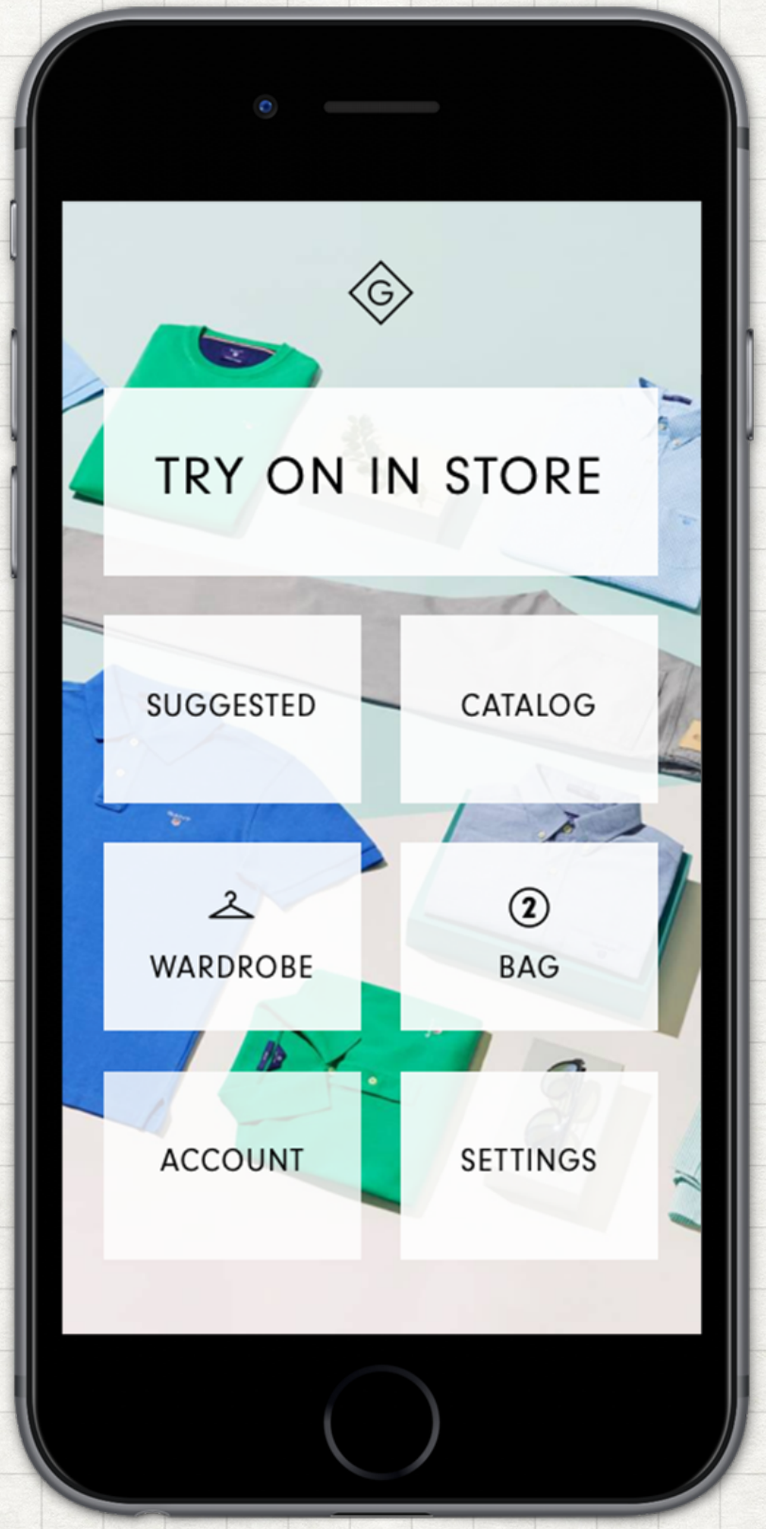
”

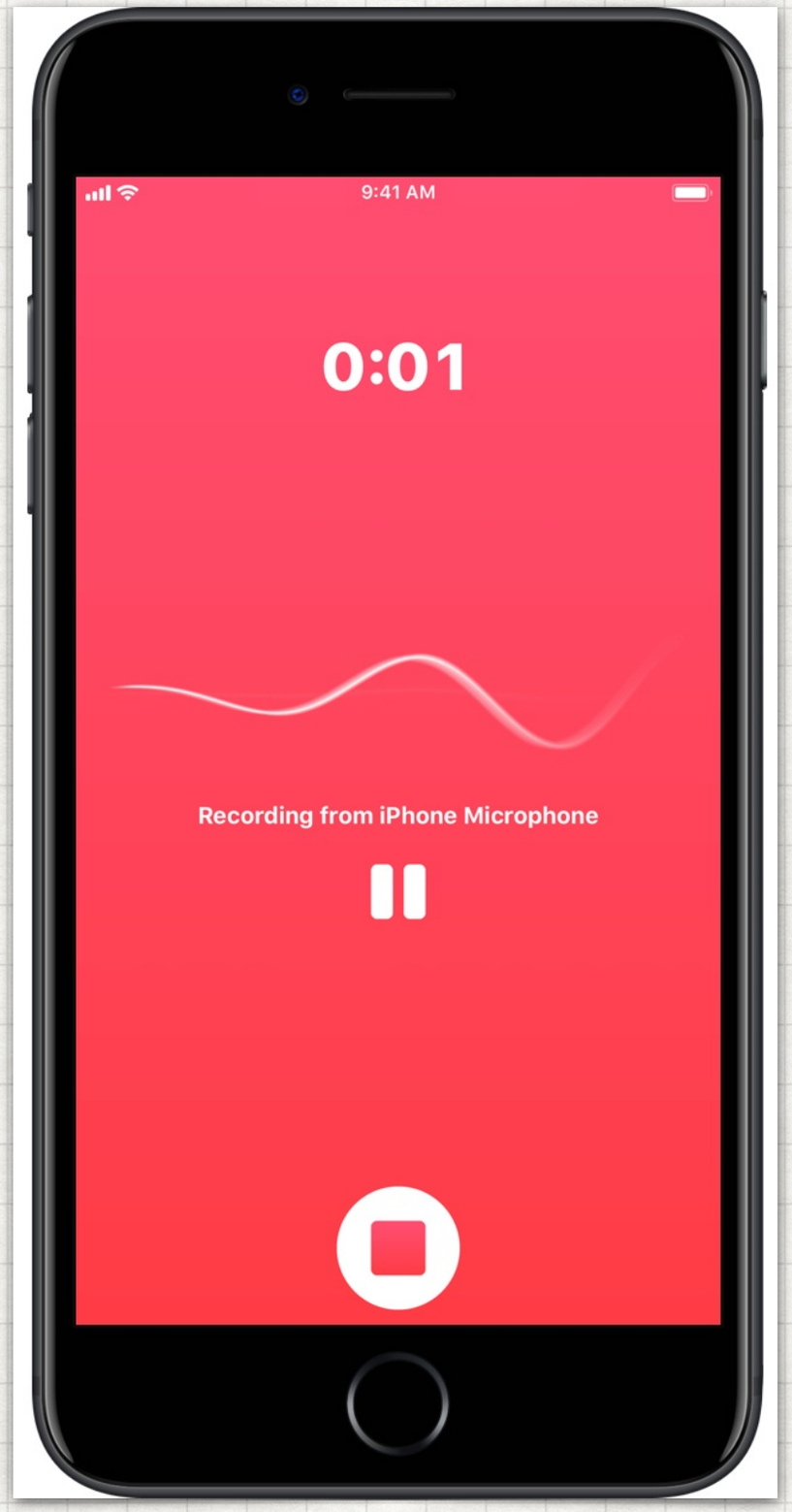
“

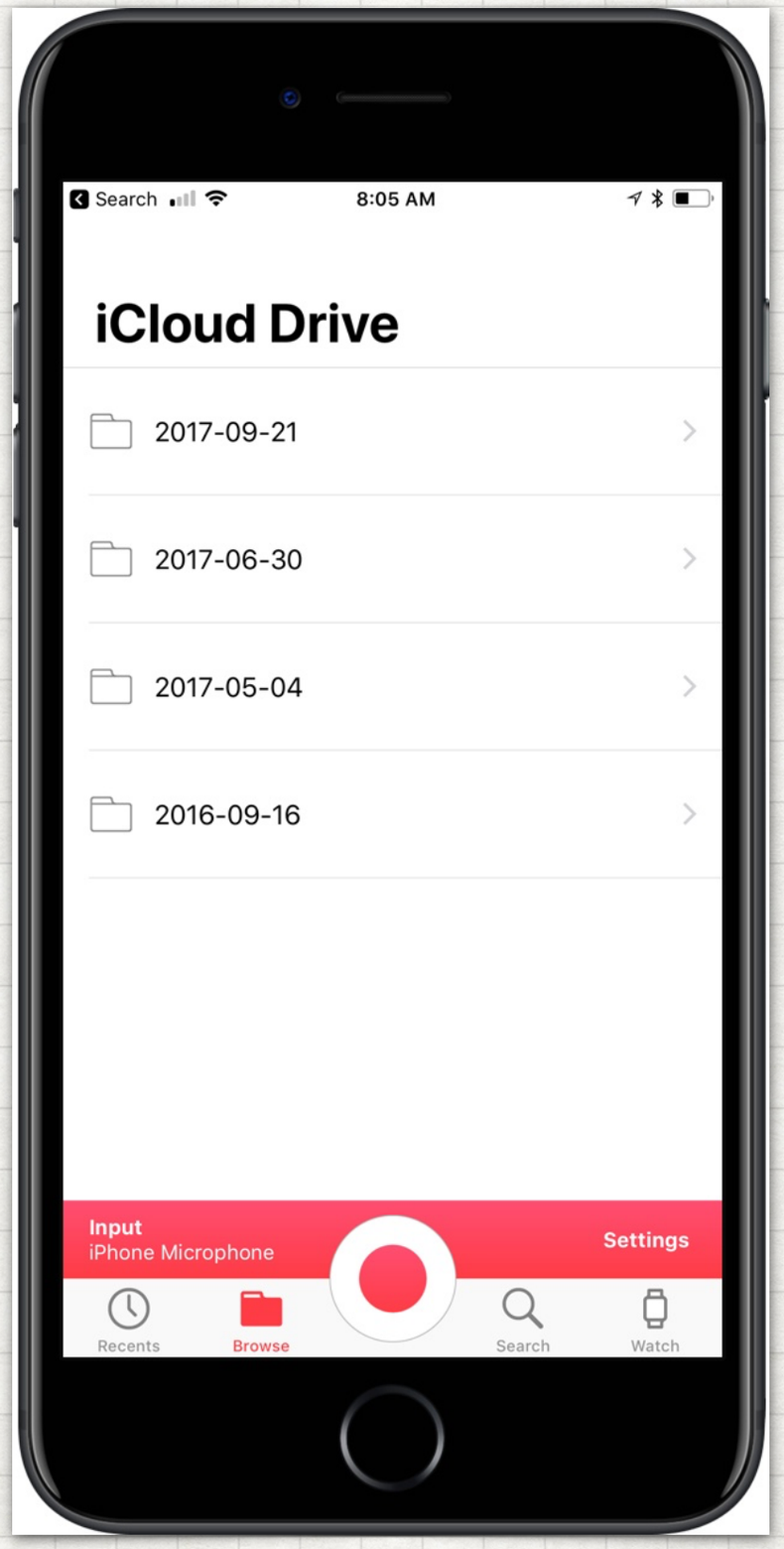
#IDEA: AN APP IS AN EXPERIENCE COMPOSED OF FLOWS

”


Verizon
82%
WHAT IS YOUR PANT SIZE?
WAIST
INSEAM
30
31
32
33
34
30
32
34
36
NOT SURE?
CONTINUE TO SHOES


TRY ON IN STORE
SUGGESTED
CATALOG
WARDROBE
BAG
ACCOUNT
SETTINGS

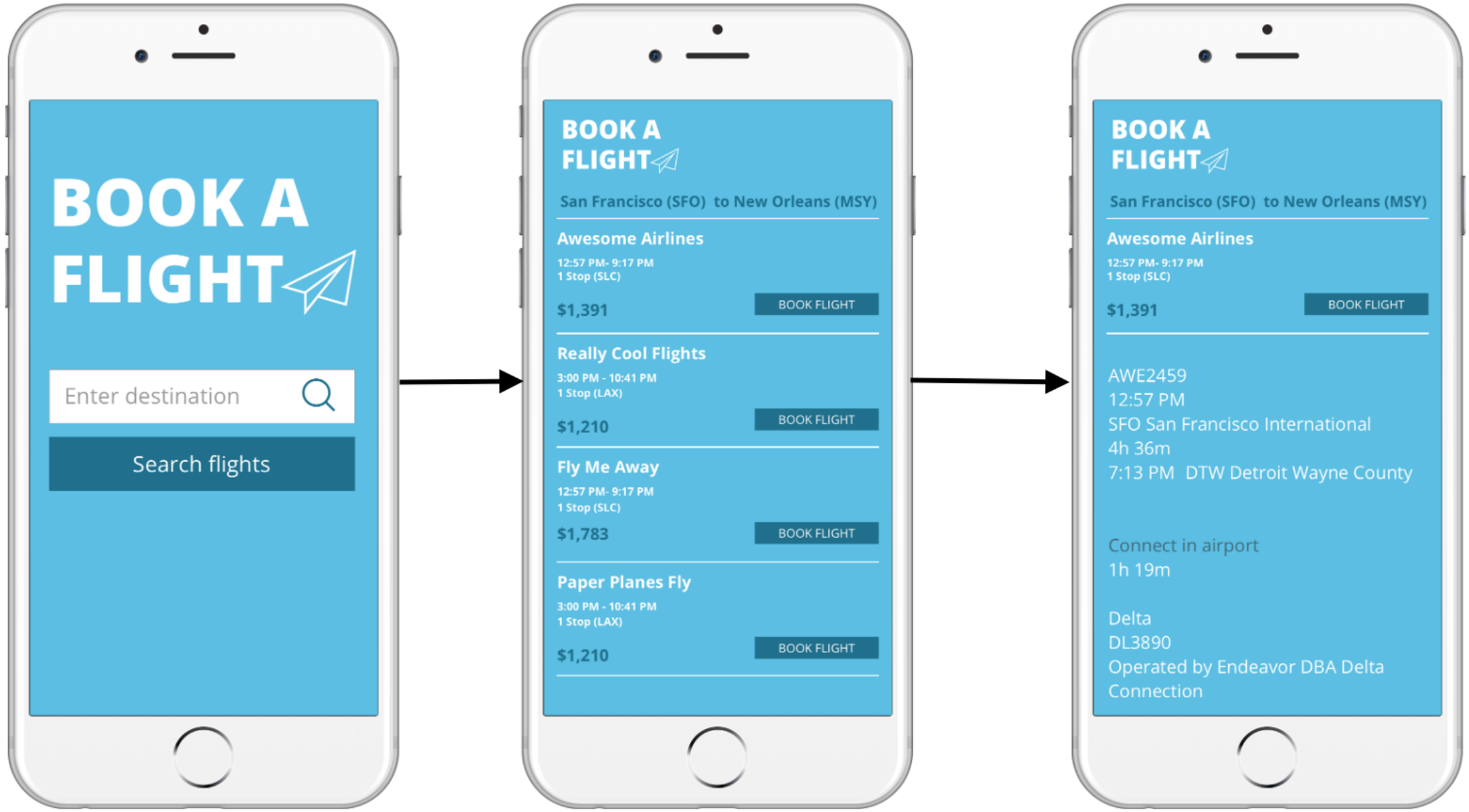

9:41 AM
0:01
Recording from iPhone Microphone


Search
8:05 AM
iCloud Drive
2017-09-21
2017-06-30
2017-05-04
2016-09-16
Input
iPhone Microphone
Settings
Recents
Browse
Search
Watch

“

A FLOW IS A SEQUENCE OF ACTIONS

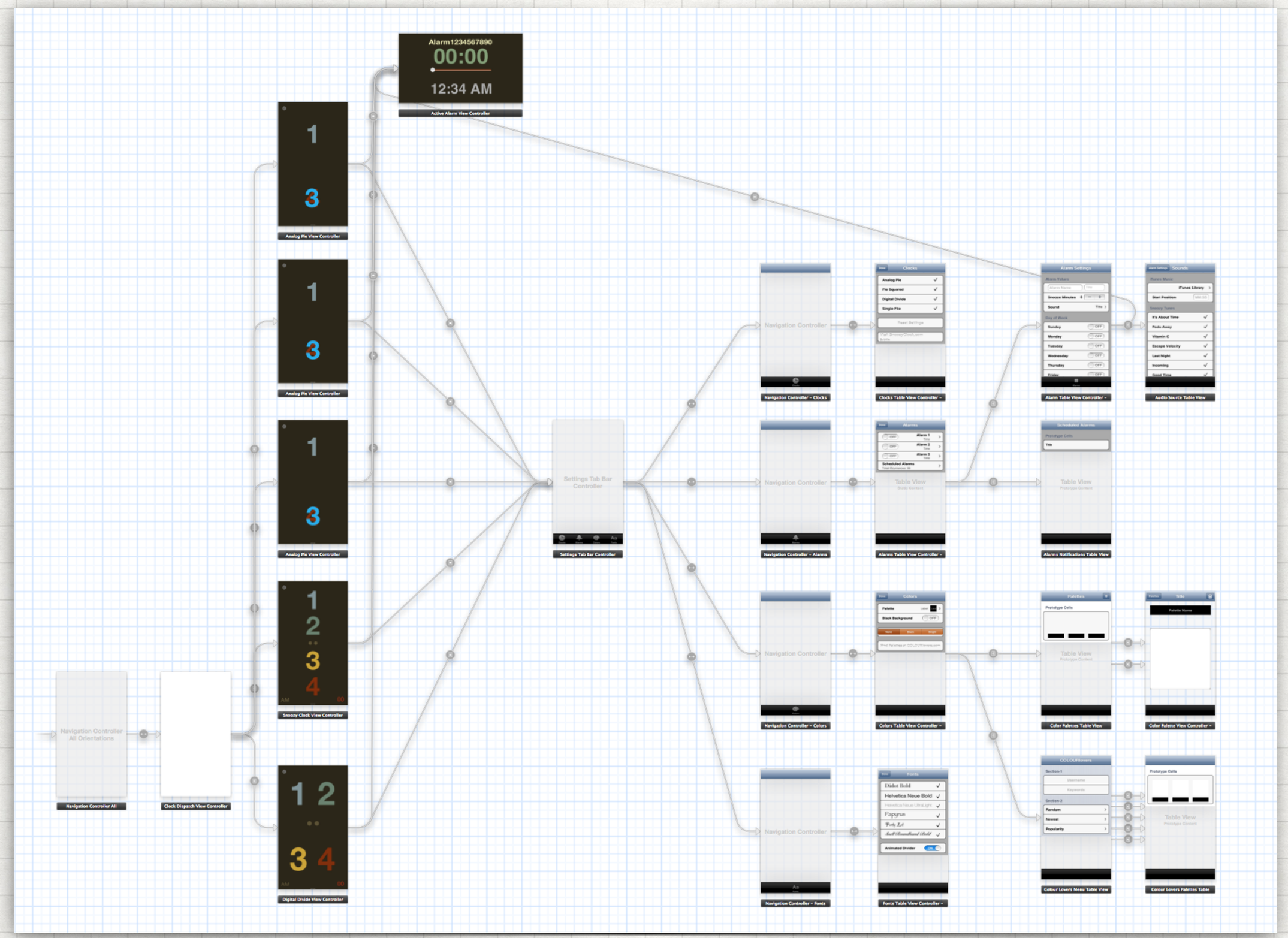
”


BOOK A FLIGHT
Enter destination
Search flights
BOOK A FLIGHT
San Francisco (SFO) to New Orleans (MSY)
Awesome Airlines
12:57 PM- 9:17 PM
1 Stop (SLC)
$1,391
BOOK FLIGHT
Really Cool Flights
3:00 PM - 10:41 PM
1 Stop (LAX)
$1,210
BOOK FLIGHT
Fly Me Away
12:57 PM- 9:17 PM
1 Stop (SLC)
$1,783
BOOK FLIGHT
Paper Planes Fly
3:00 PM - 10:41 PM
1 Stop (LAX)
$1,210
BOOK FLIGHT
BOOK A FLIGHT
San Francisco (SFO) to New Orleans (MSY)
Awesome Airlines
12:57 PM- 9:17 PM
1 Stop (SLC)
$1,391
BOOK FLIGHT
AWE2459
12:57 PM
SFO San Francisco International
4h 36m
7:13 PM DTW Detroit Wayne County
Connect in airport
1h 19m
Delta
DL3890
Operated by Endeavor DBA Delta Connection

“

FLOWS ARE EASY TO VISUALLY MODEL

”


Alarm1234567890
00:00
12:34 AM
Active Alarm View Controller
Analog Pie View Controller
Analog Pie View Controller
Analog Pie View Controller
Snoozy Clock View Controller
Digital Divide View Controller
Navigation Controller All Orientations
Navigation Controller All
Clock Dispatch View Controller
Settings Tab Bar Controller
Settings Tab Bar Controller
Navigation Controller
Navigation Controller - Clocks
Clocks
Clocks Table View Controller -
Alarm Settings
Alarm Table View Controller -
Sounds
Audio Source Table View
Navigation Controller - Alarms
Alarms
Alarms Table View Controller -
Scheduled Alarms
Alarms Notifications Table View
Navigation Controller - Colors
Colors
Colors Table View Controller -
Palettes
Color Palettes Table View
Title
Color Palette View Controller -
Navigation Controller - Fonts
Fonts
Fonts Table View Controller -
COLOURlovers
Colour Lovers Menu Table View
Colour Lovers Palettes Table

“

CAN OUR ARCHITECTURE MATCH OUR FLOWS? YES!

”

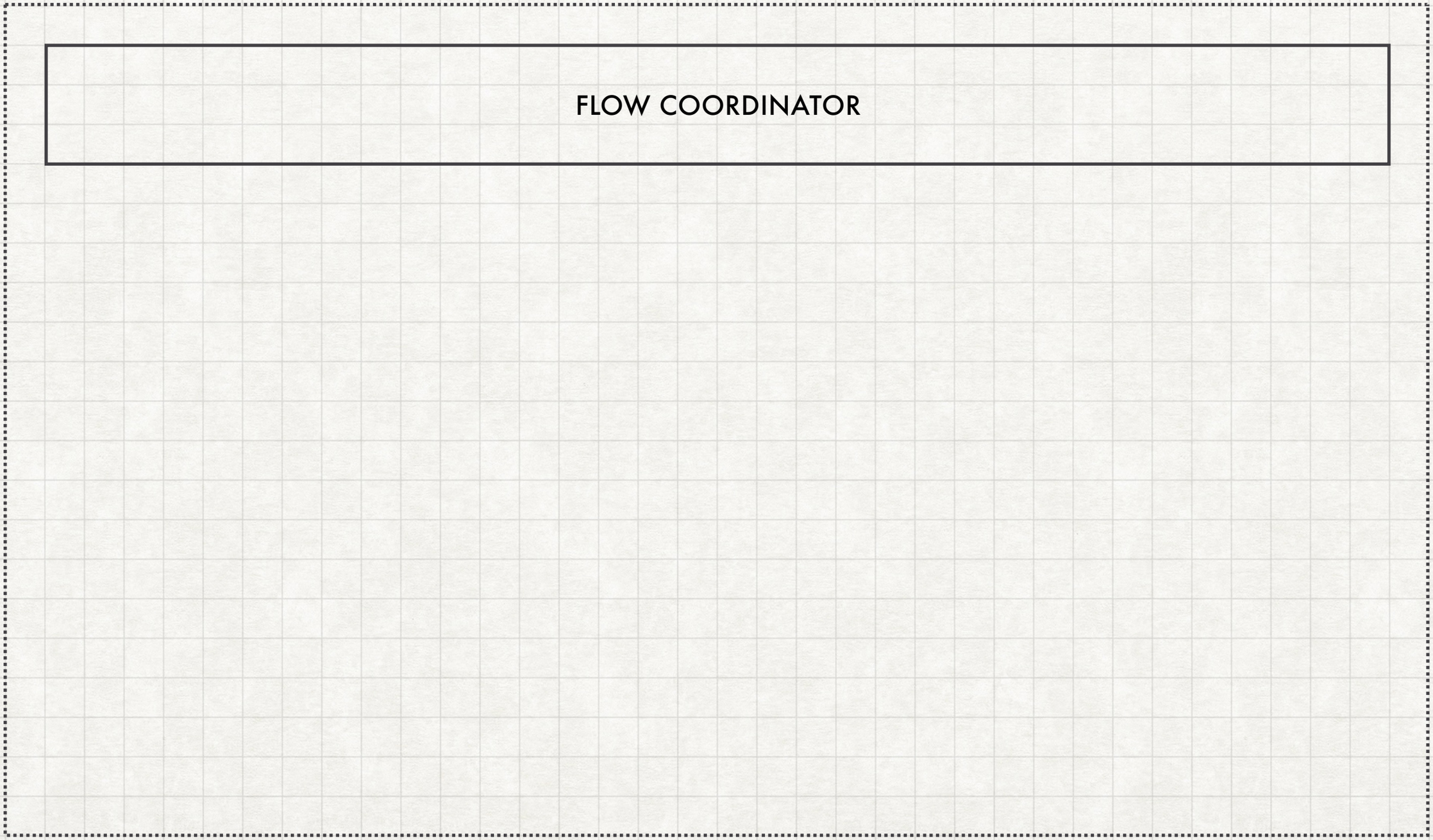
COMPONENTS

OF A "FLOW-BASED" ARCHITECTURE

- Flow Coordinators (FCs) - "manage the flow"
- View Controllers (VCs) - "take user input, propagate data"
- Data Sources (DSs) - "get data from network, disc, … wherever"
- Views - "display all the things!"
- Models - "represent all the things!"

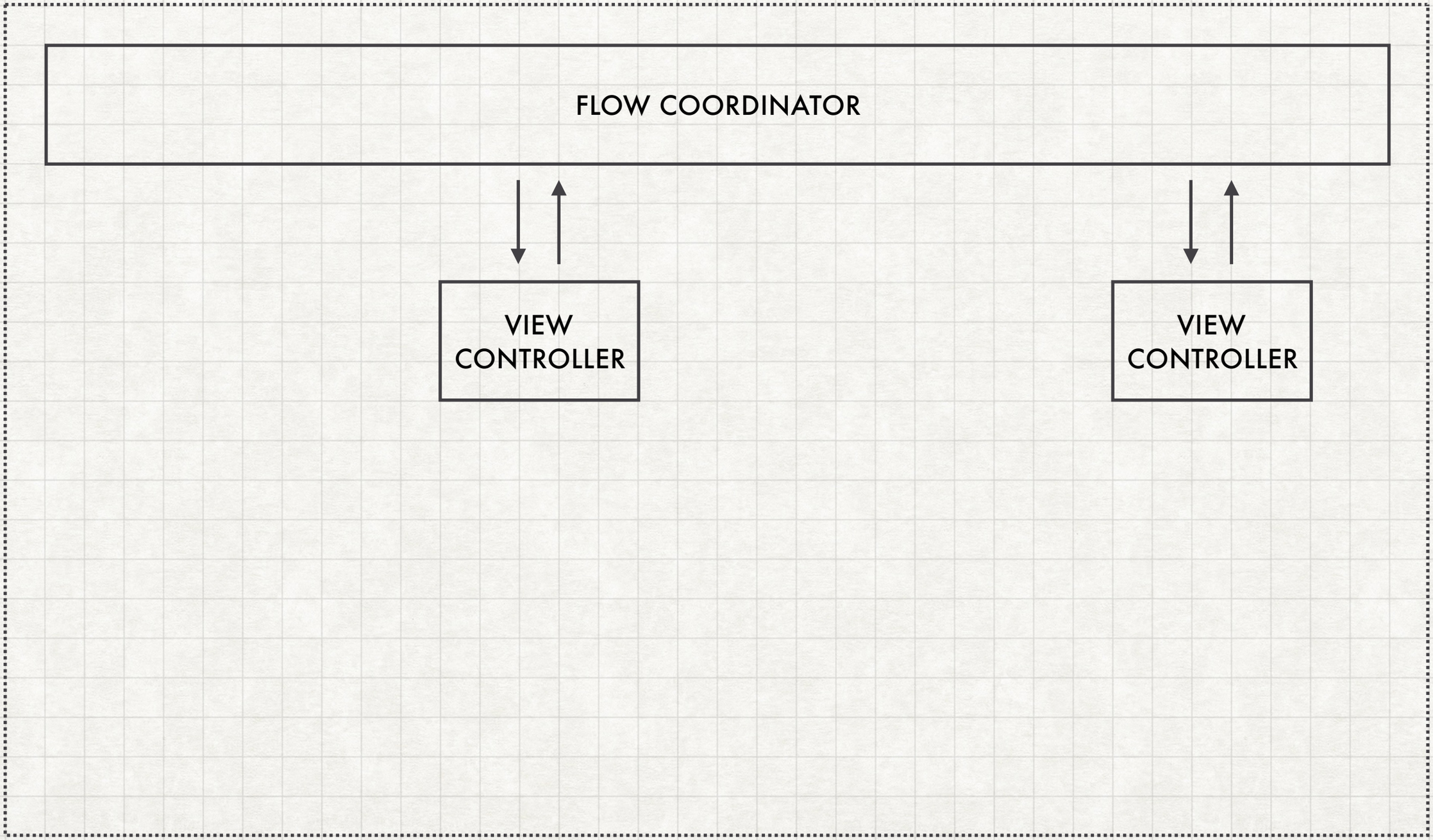
COMPONENTS

OF A SINGLE FLOW



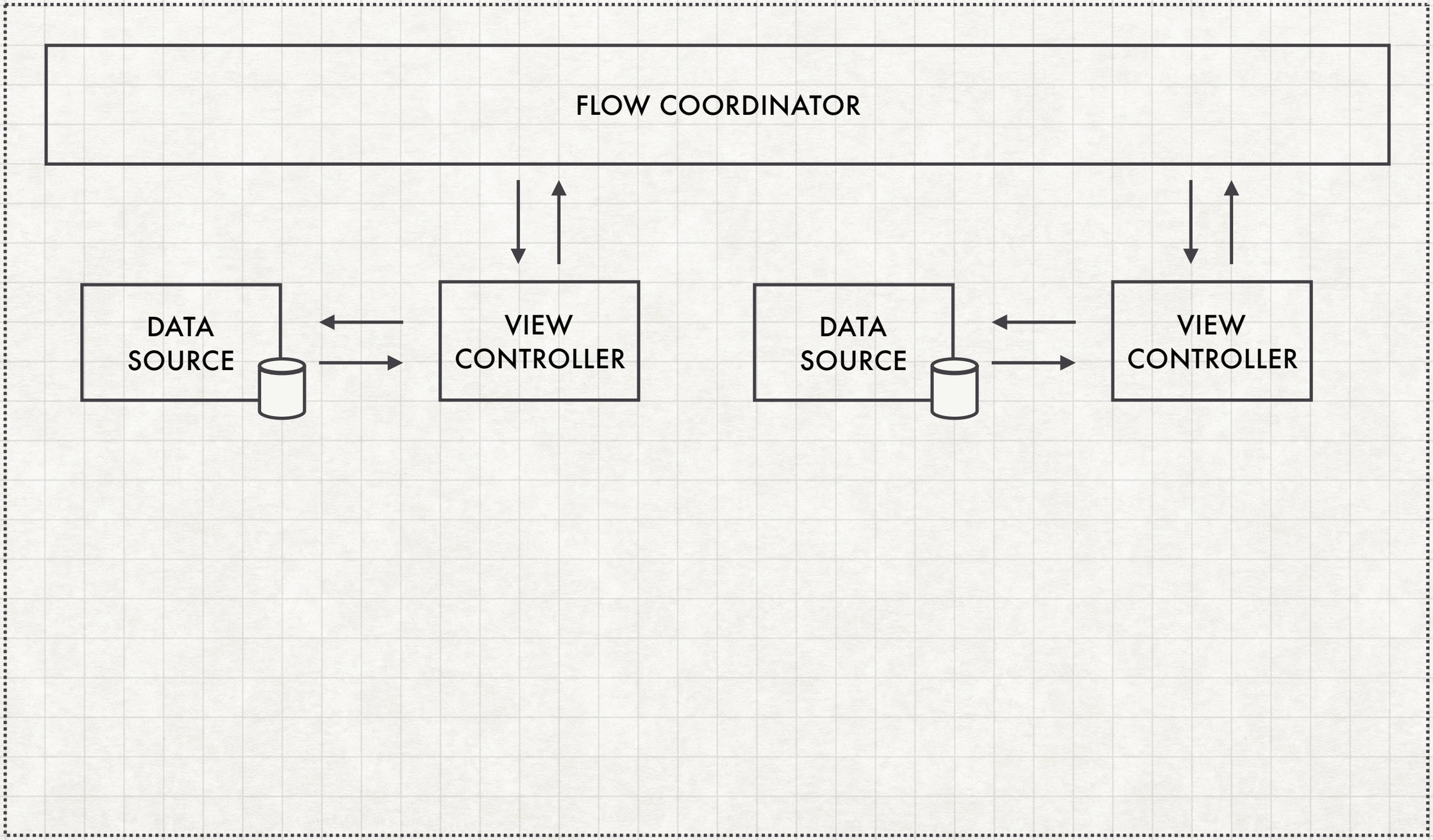
COMPONENTS

OF A SINGLE FLOW



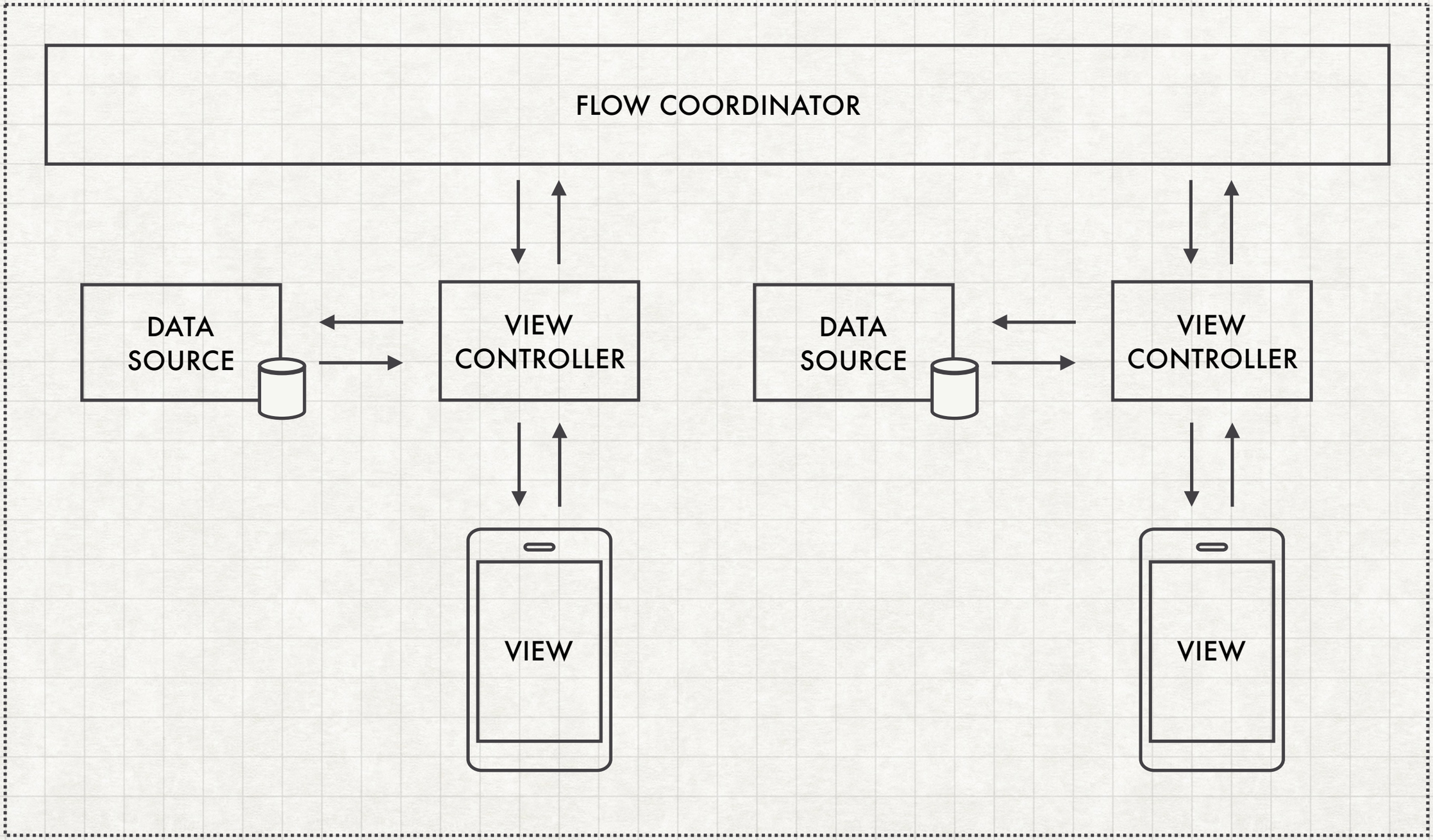
COMPONENTS

OF A SINGLE FLOW



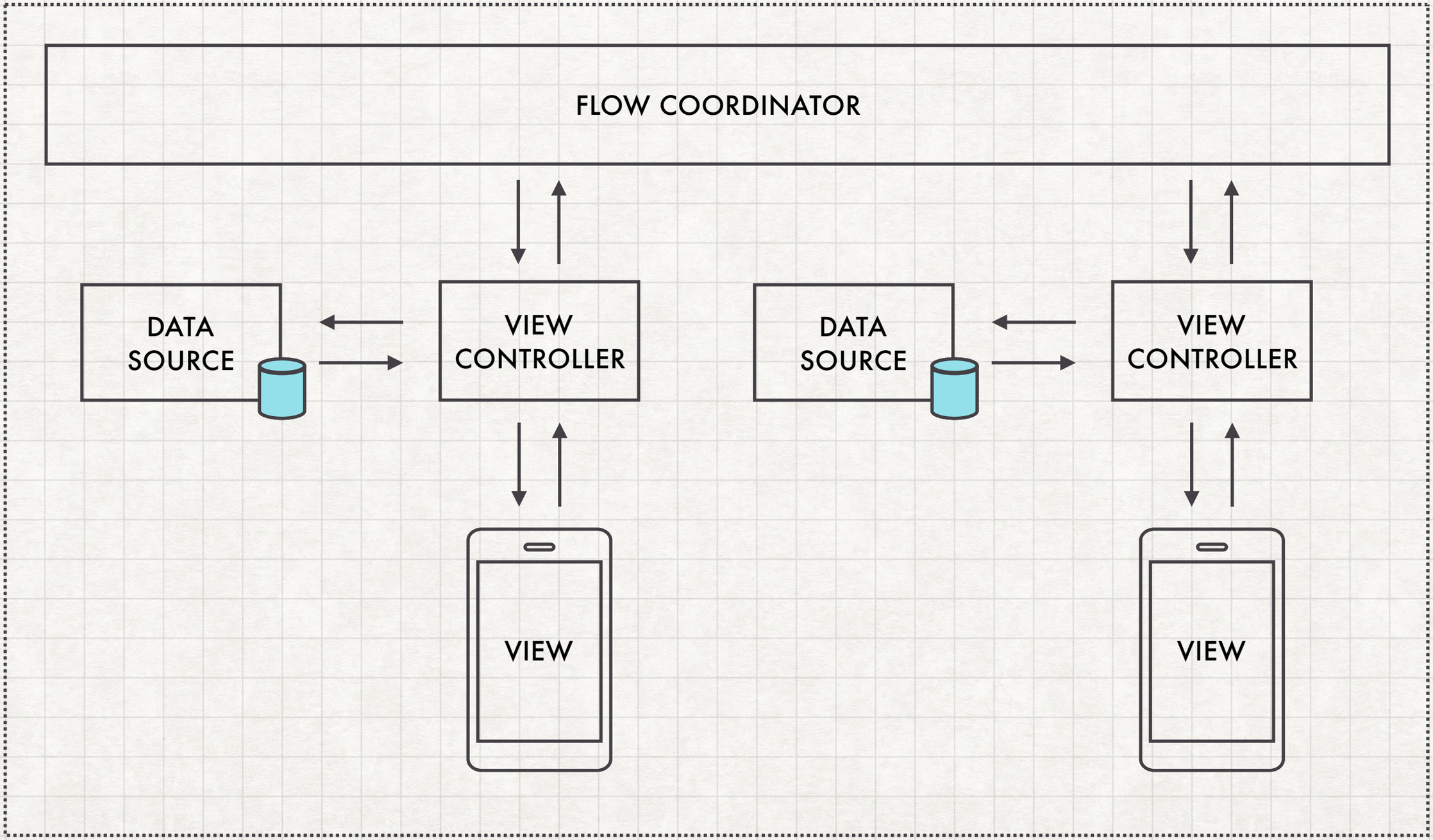
COMPONENTS

OF A SINGLE FLOW



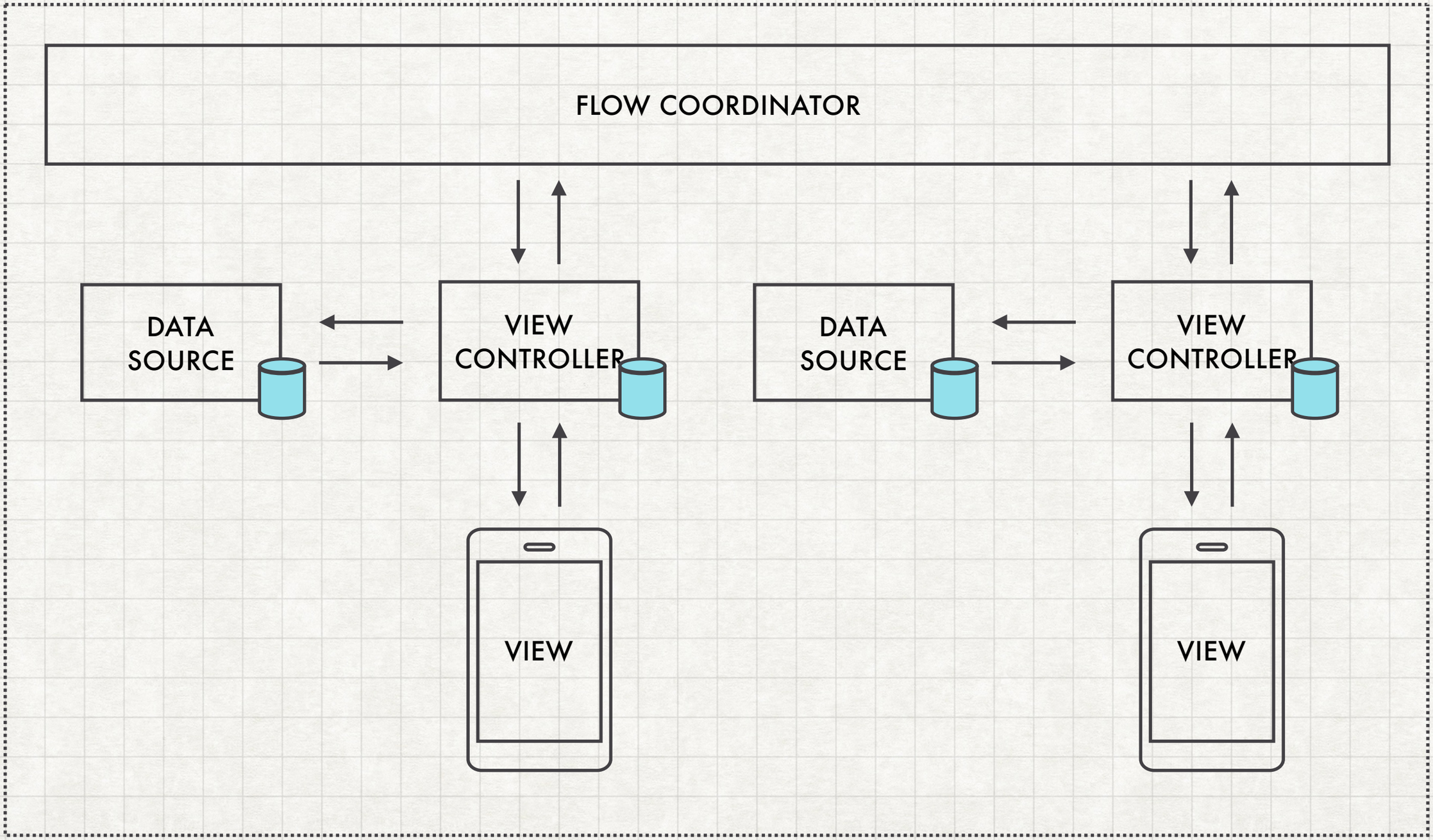
COMPONENTS

OF A SINGLE FLOW



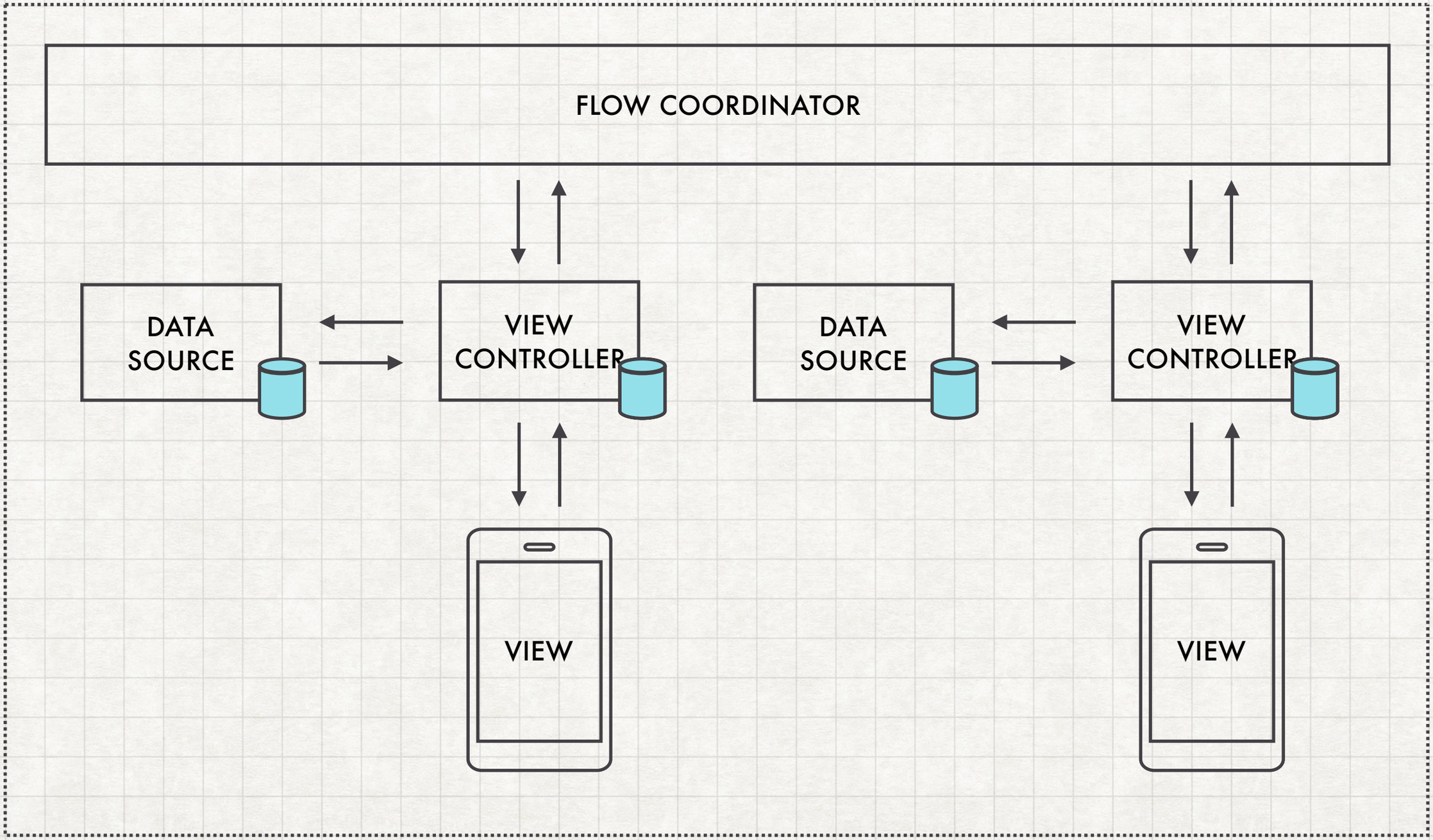
COMPONENTS

OF A SINGLE FLOW



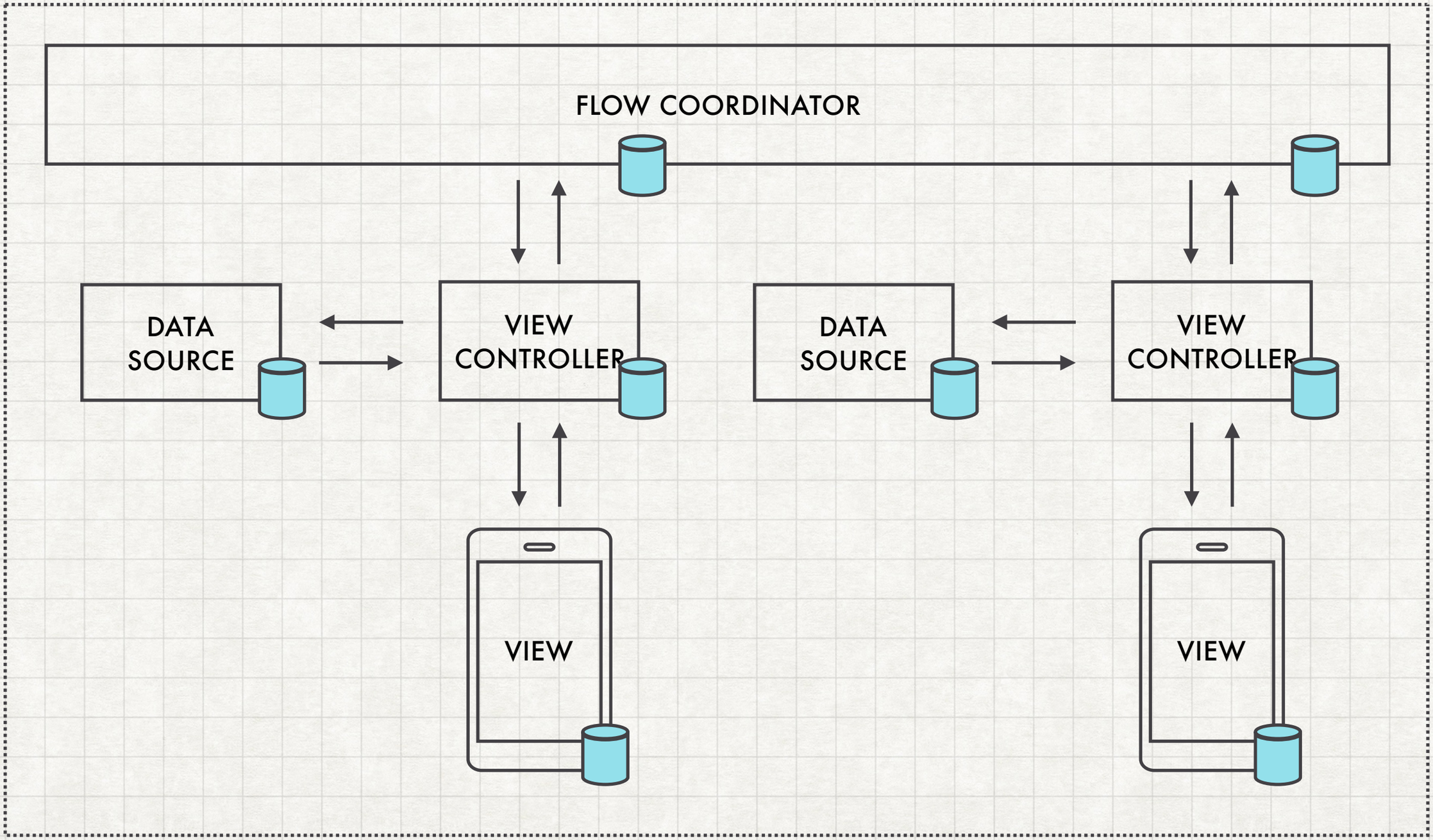
COMPONENTS

OF A SINGLE FLOW



COMPONENTS

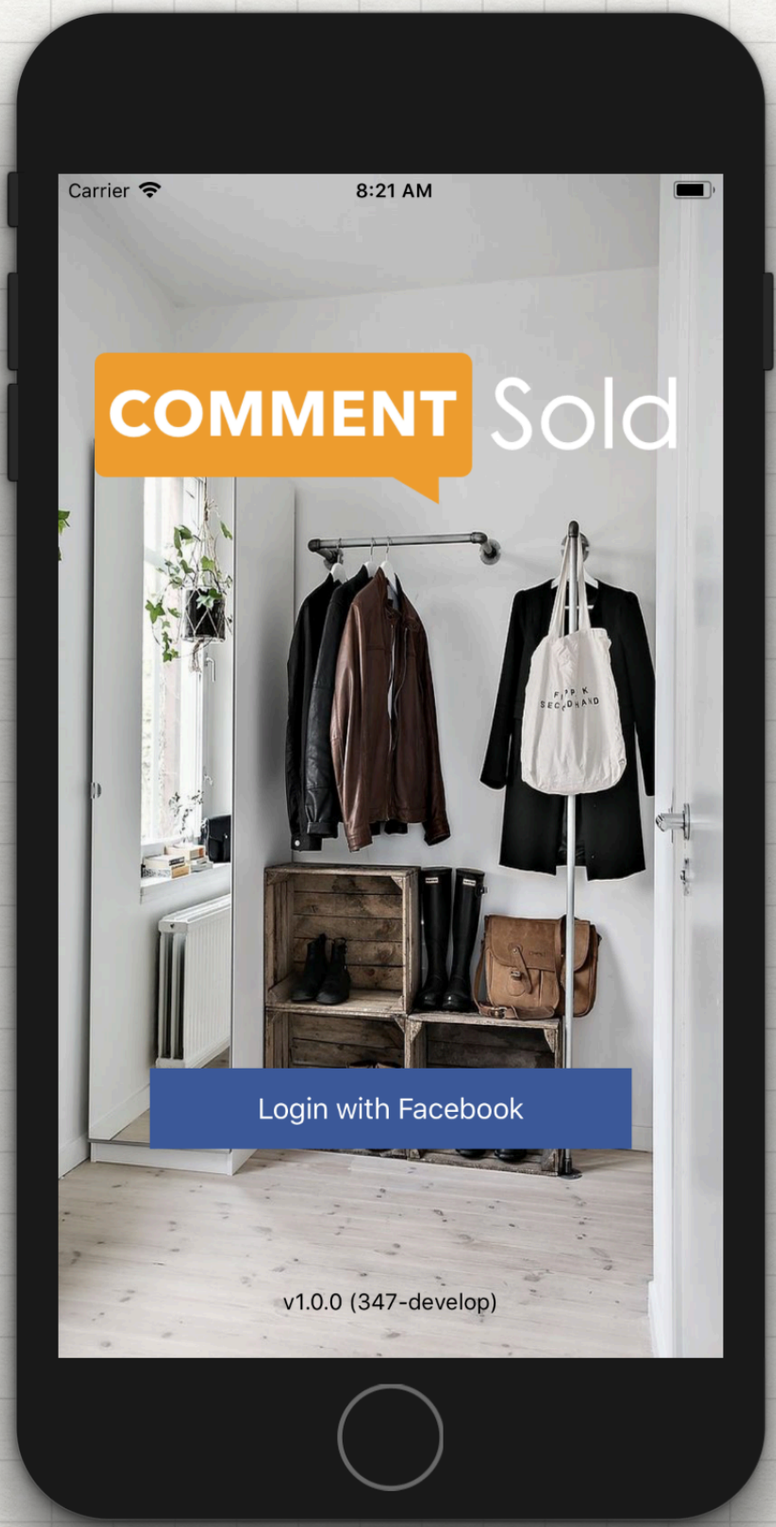
OF A SINGLE FLOW



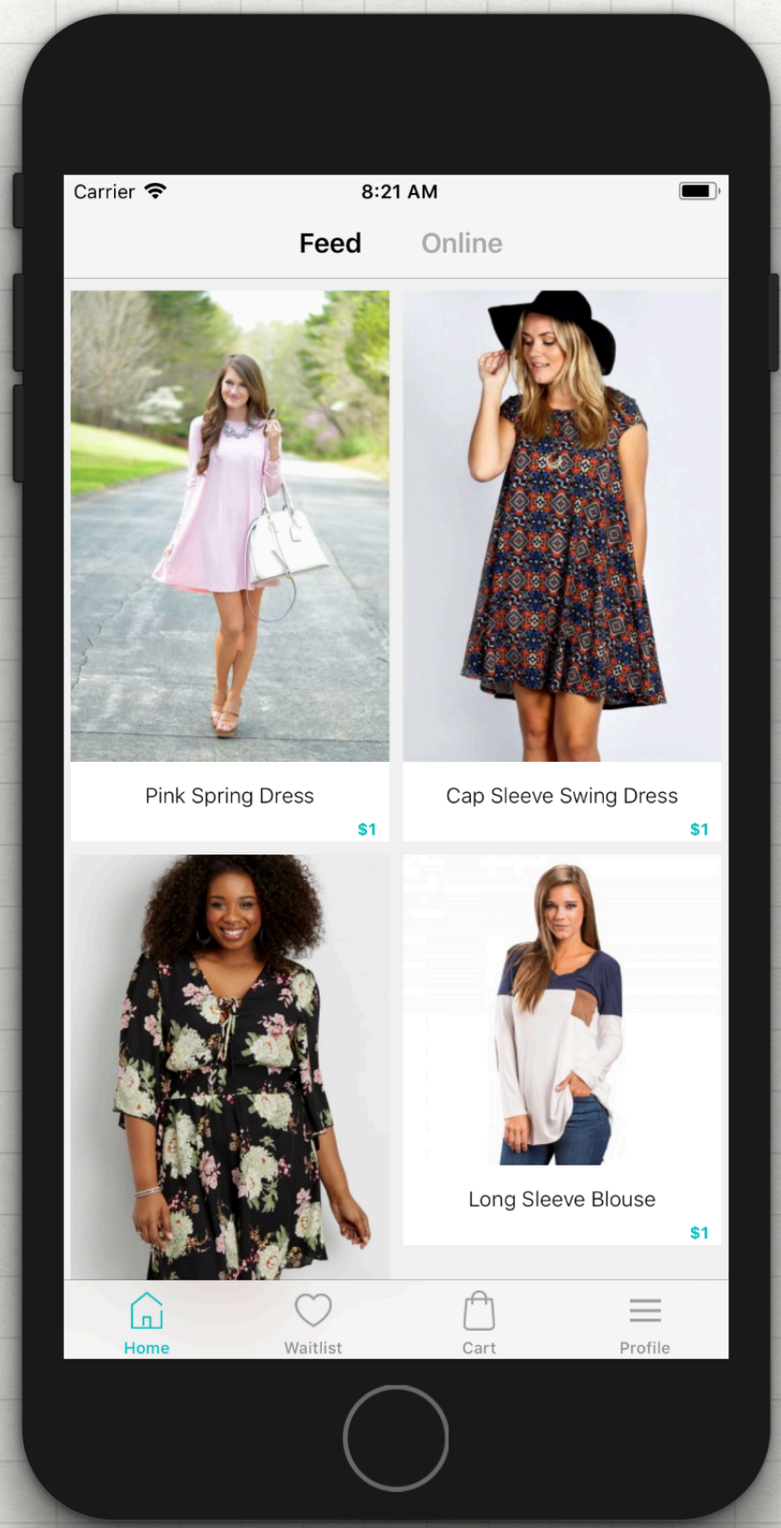
“

LET'S LOOK AT AN EXAMPLE!

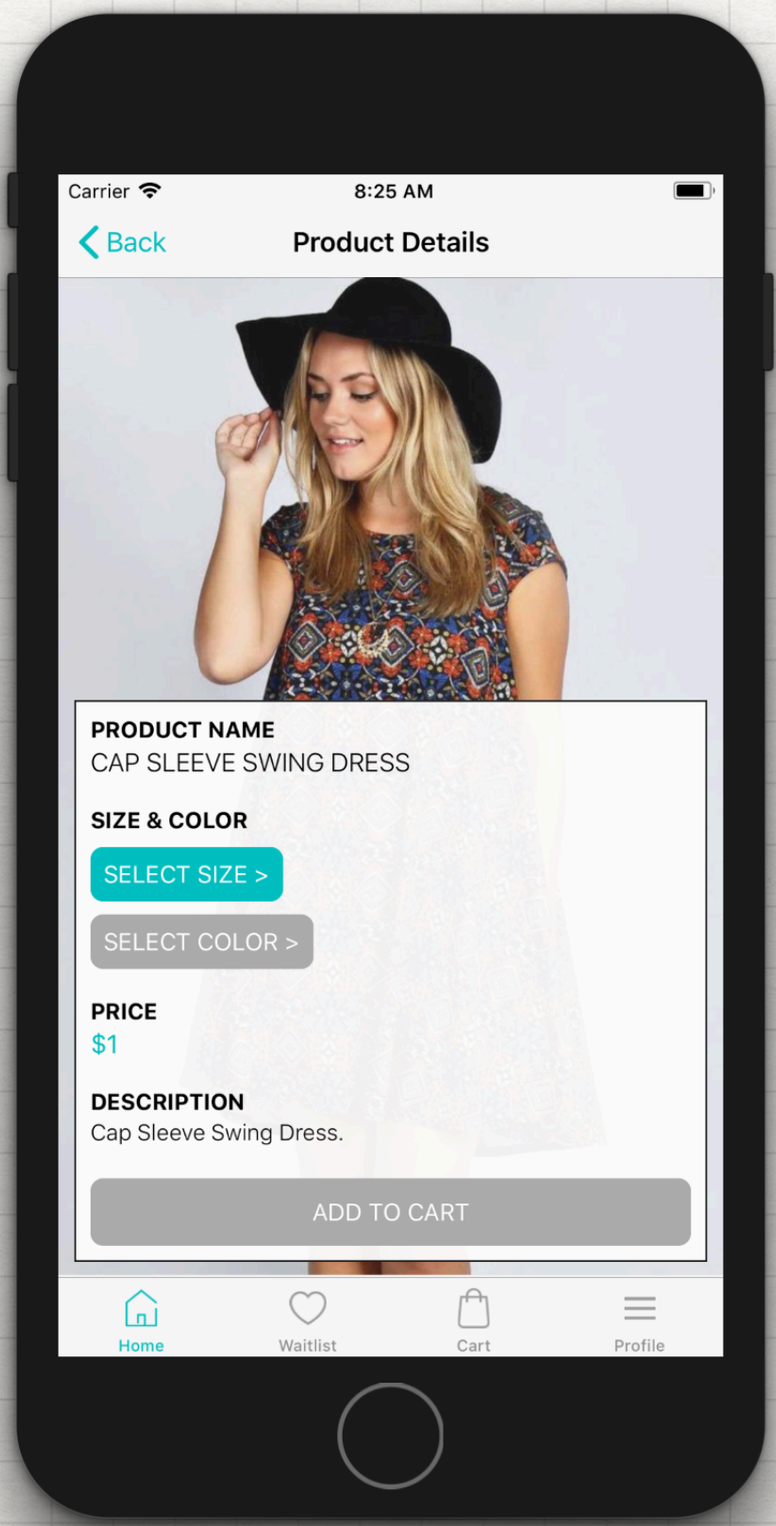
”



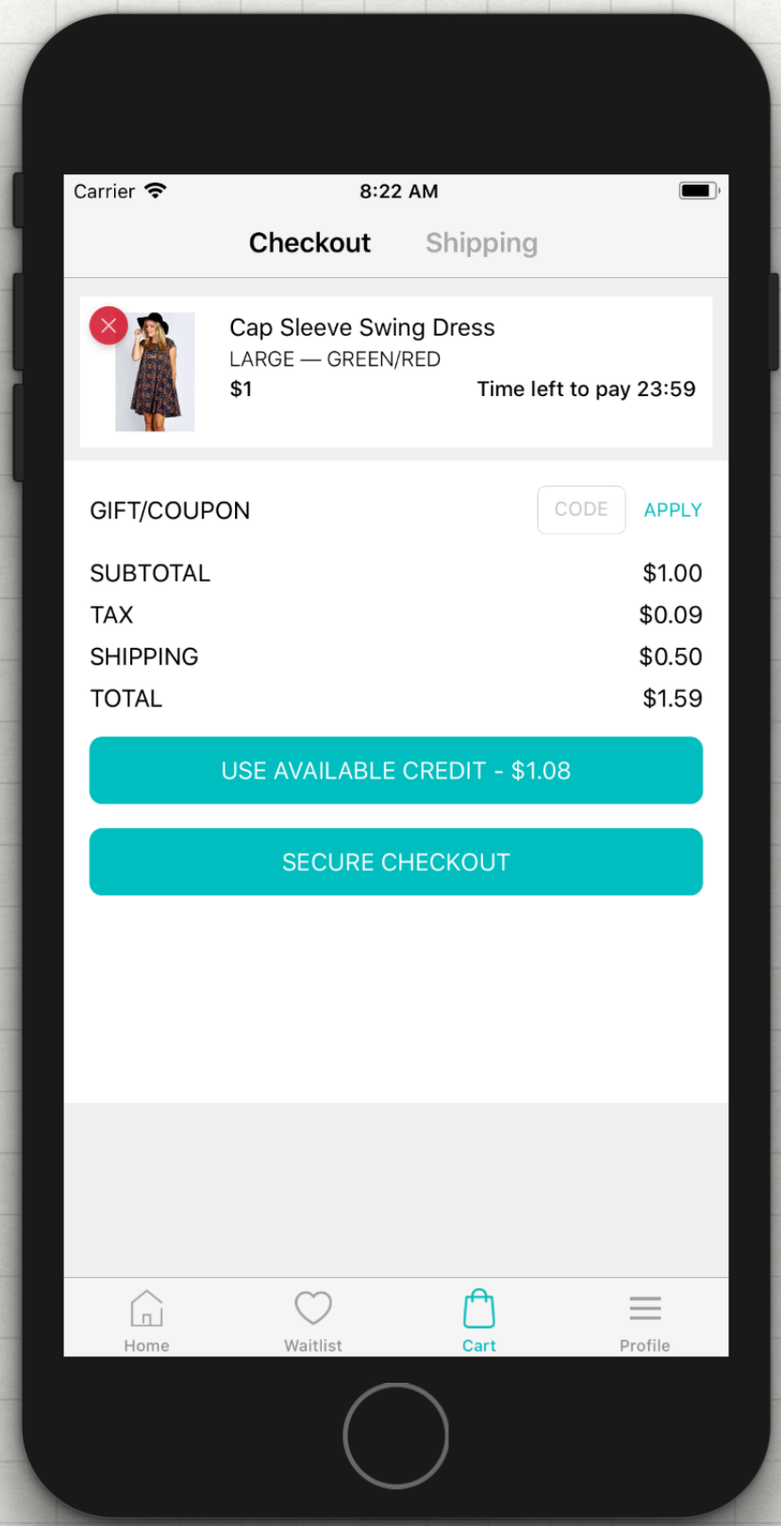
iPhone 6 Plus - 11.2



iPhone 6 Plus - 11.2

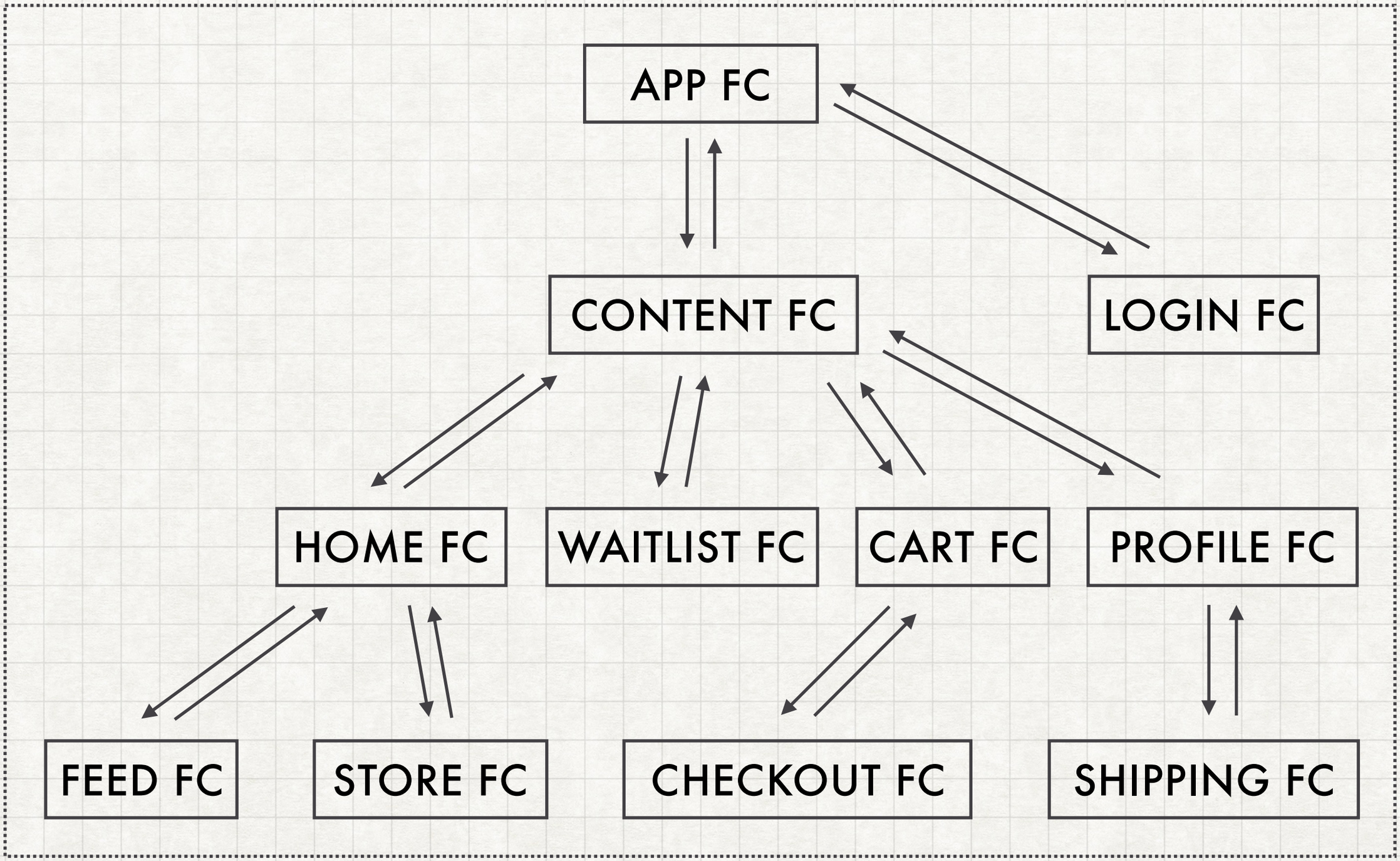


iPhone 6 Plus - 11.2



iPhone 6 Plus - 11.2

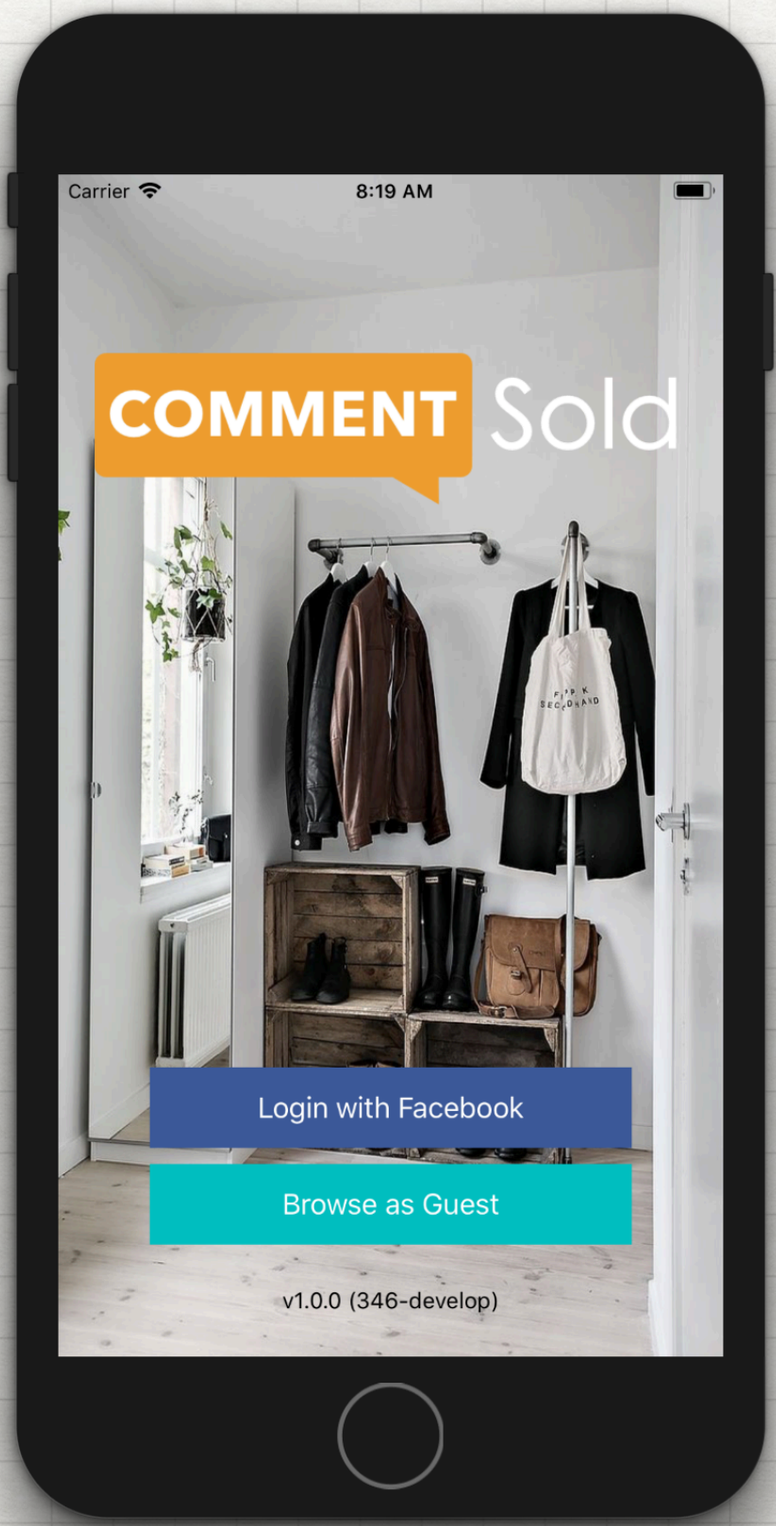
COMMENT SOLD FLOWS



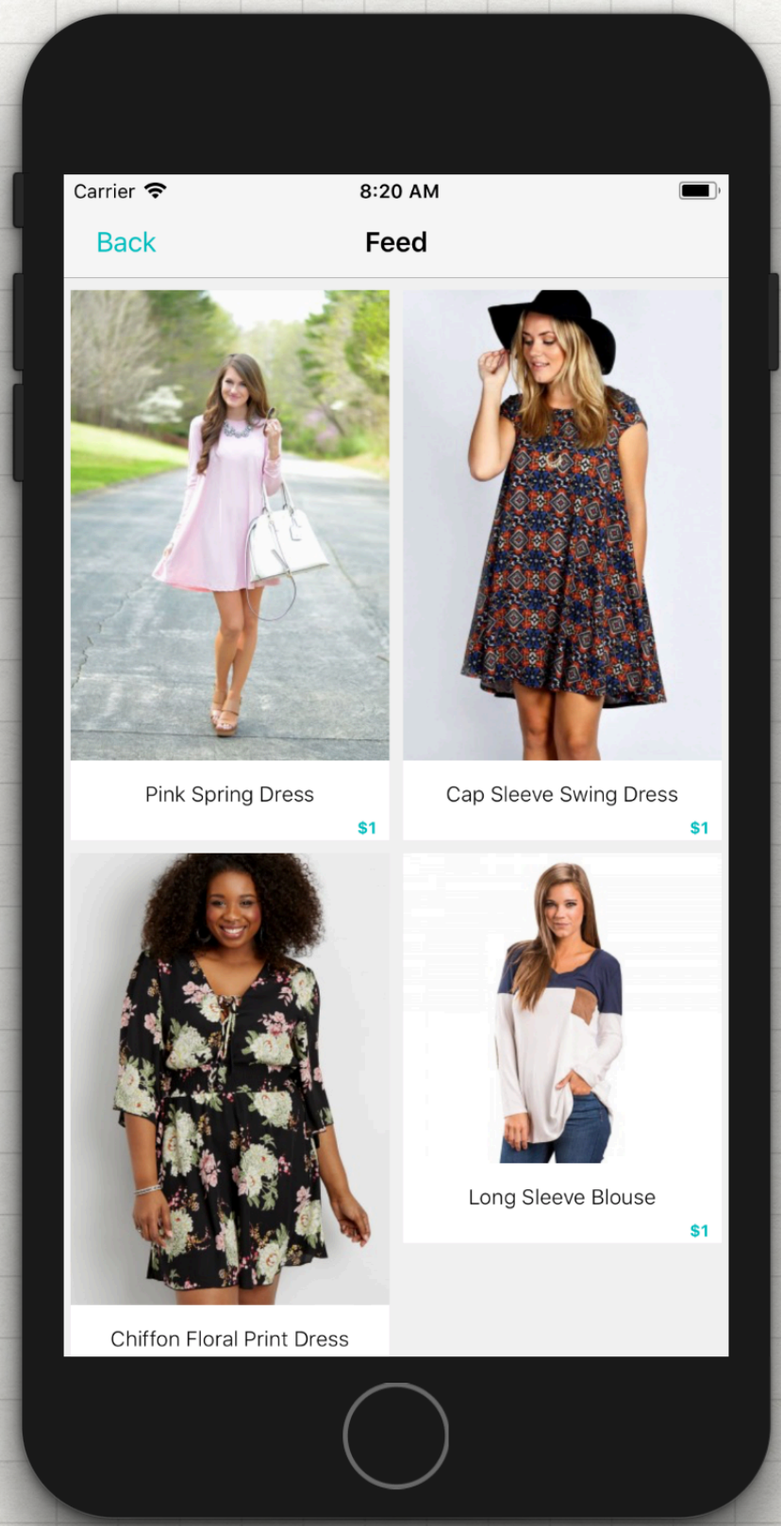
“

ADD A FLOW. SIMPLE, RIGHT?

”

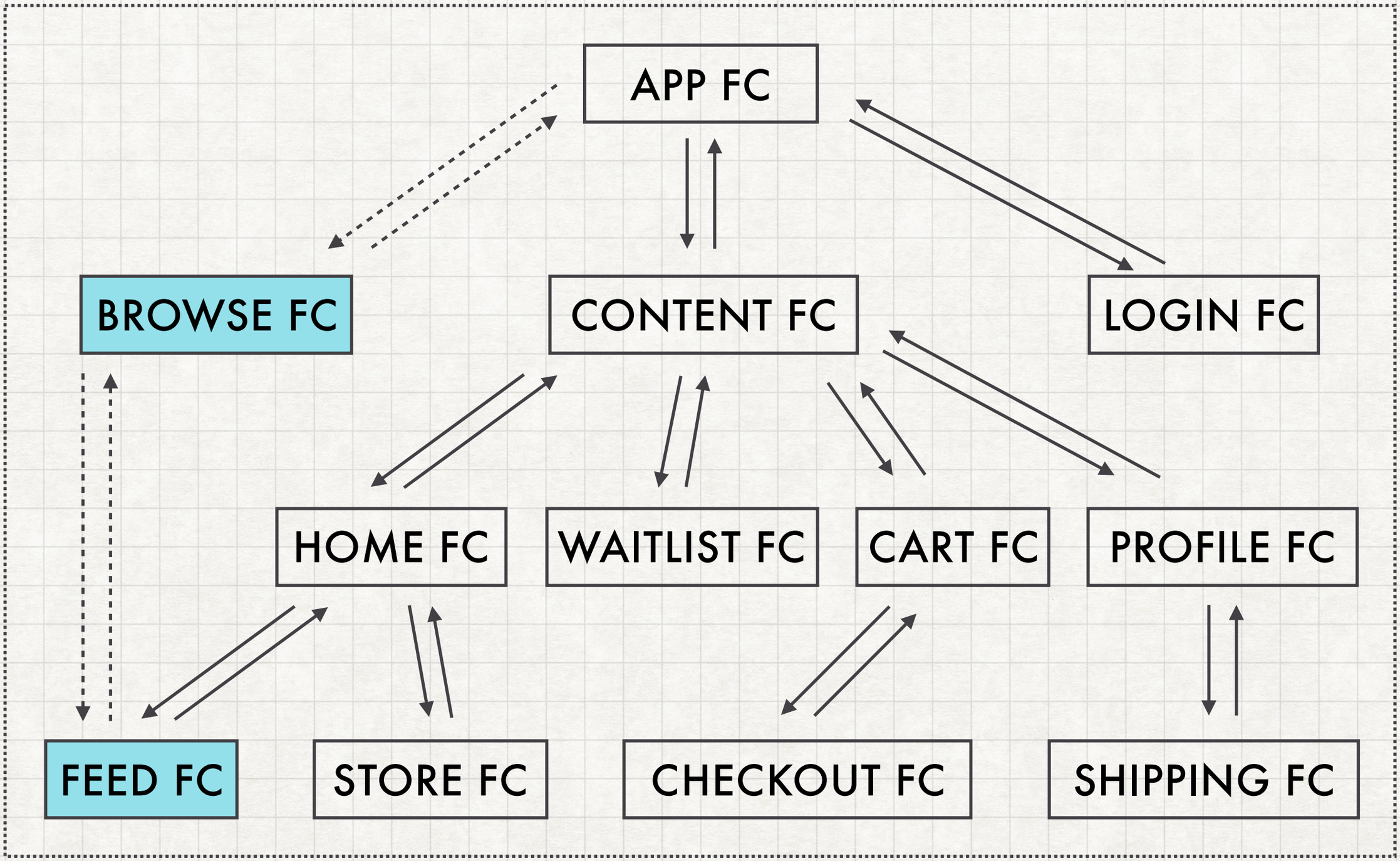


iPhone 6 Plus - 11.2



iPhone 6 Plus - 11.2

COMMENT SOLD FLOWS



“

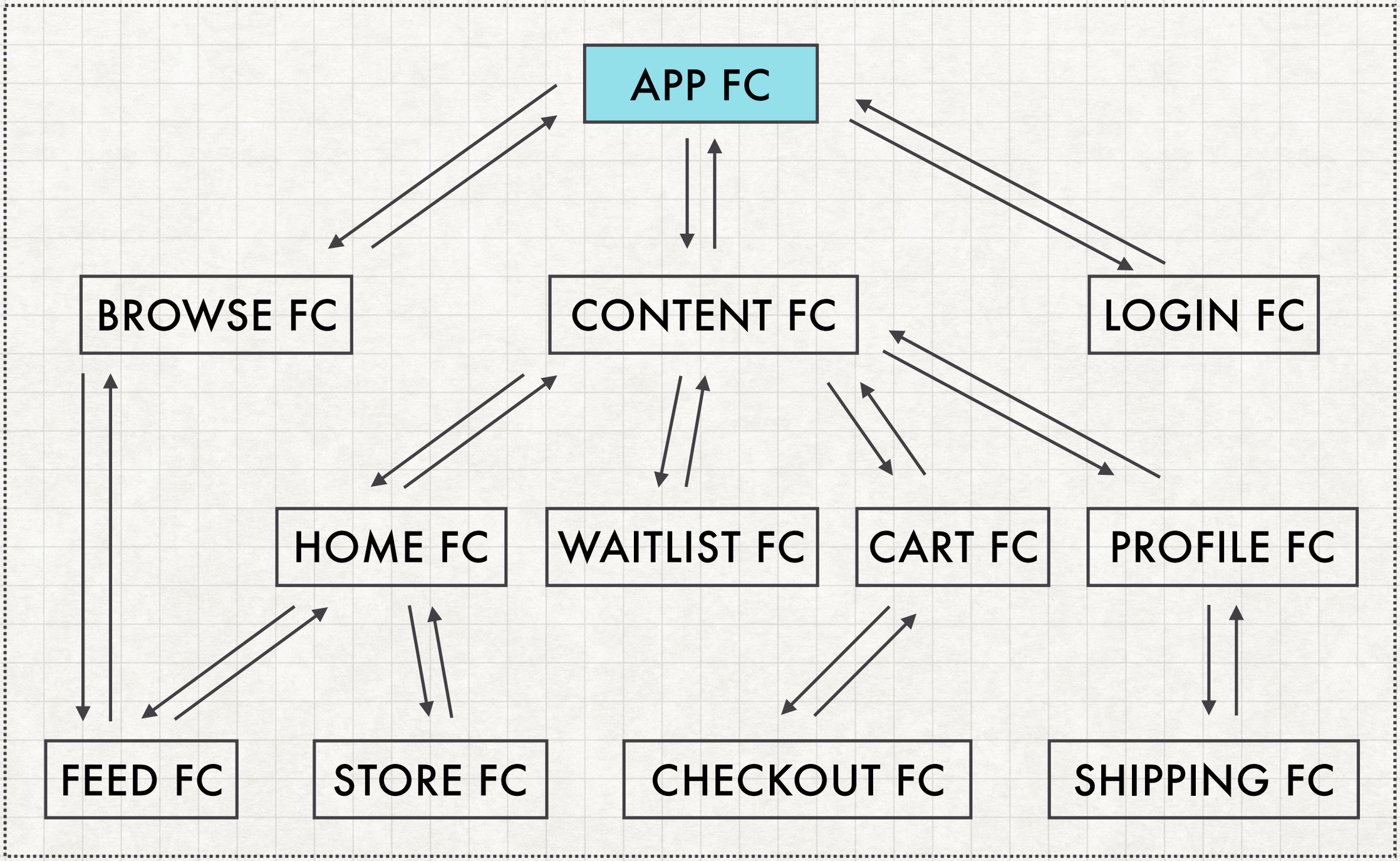
OKAY. WHAT ABOUT THE CODE? IT'S UGLY, RIGHT?

”

```
// MARK: - FlowCoordinator

public protocol FlowCoordinator: class {
    var services: Services { get }
    var childFCs: [FlowCoordinator] { get set }
}
```

```swift
// MARK: - RootVCProvider

public protocol RootVCProvider: class {
    var rootViewController: UIViewController { get }
}

public typealias RootViewFC = FlowCoordinator & RootVCProvider
```

COMMENT SOLD FLOWS



```
public class AppFC: RootViewFC {

    // MARK: Properties

    public let services: Services
    public var childFCs: [FlowCoordinator] = []
    public var rootViewController: UIViewController { return navigationController }

    private let window: UIWindow
    private lazy var navigationController: UINavigationController = {
        let navigationController = UINavigationController()
        navigationController.isNavigationBarHidden = true
        return navigationController
    }()

    // MARK: Initializer

    public init(window: UIWindow, services: Services) {
        self.services = services
        self.window = window

        window.rootViewController = rootViewController
        window.makeKeyAndVisible()
    }

    // ...
}
```

```
var appFC: AppFC!

func application(_ application: UIApplication,
            didFinishLaunchingWithOptions launchOptions:
            [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

    // prepare window
    window = UIWindow(frame: UIScreen.main.bounds)

    // create app coordinator and start!
    appFC = AppFC(window: window!, services: configureServices())

    if let launchOptions = launchOptions,
       let userInfo = launchOptions[.remoteNotification],
       let notification = CSNotification(userInfo: userInfo) {
       appFC.start(withNotification: notification)
    } else {
       appFC.start()
    }

    return true
}
```

```
var appFC: AppFC!

func application(_ application: UIApplication,
            didFinishLaunchingWithOptions launchOptions:
            [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

    // prepare window
    window = UIWindow(frame: UIScreen.main.bounds)

    // create app coordinator and start!
    appFC = AppFC(window: window!, services: configureServices())

    if let launchOptions = launchOptions,
       let userInfo = launchOptions[.remoteNotification],
       let notification = CSNotification(userInfo: userInfo) {
       appFC.start(withNotification: notification)
    } else {
       appFC.start()
    }

    return true
}
```

“

ALL SYSTEM EVENTS CAN BE PROPAGATED THROUGH THE APP FLOW COORDINATOR

”

```
// in AppFC...

public func start() {
    autoLogin() { (error) in
        if let error = error {
            self.startLoginFC()
        } else {
            self.startContentFC()
        }
    }
}
```

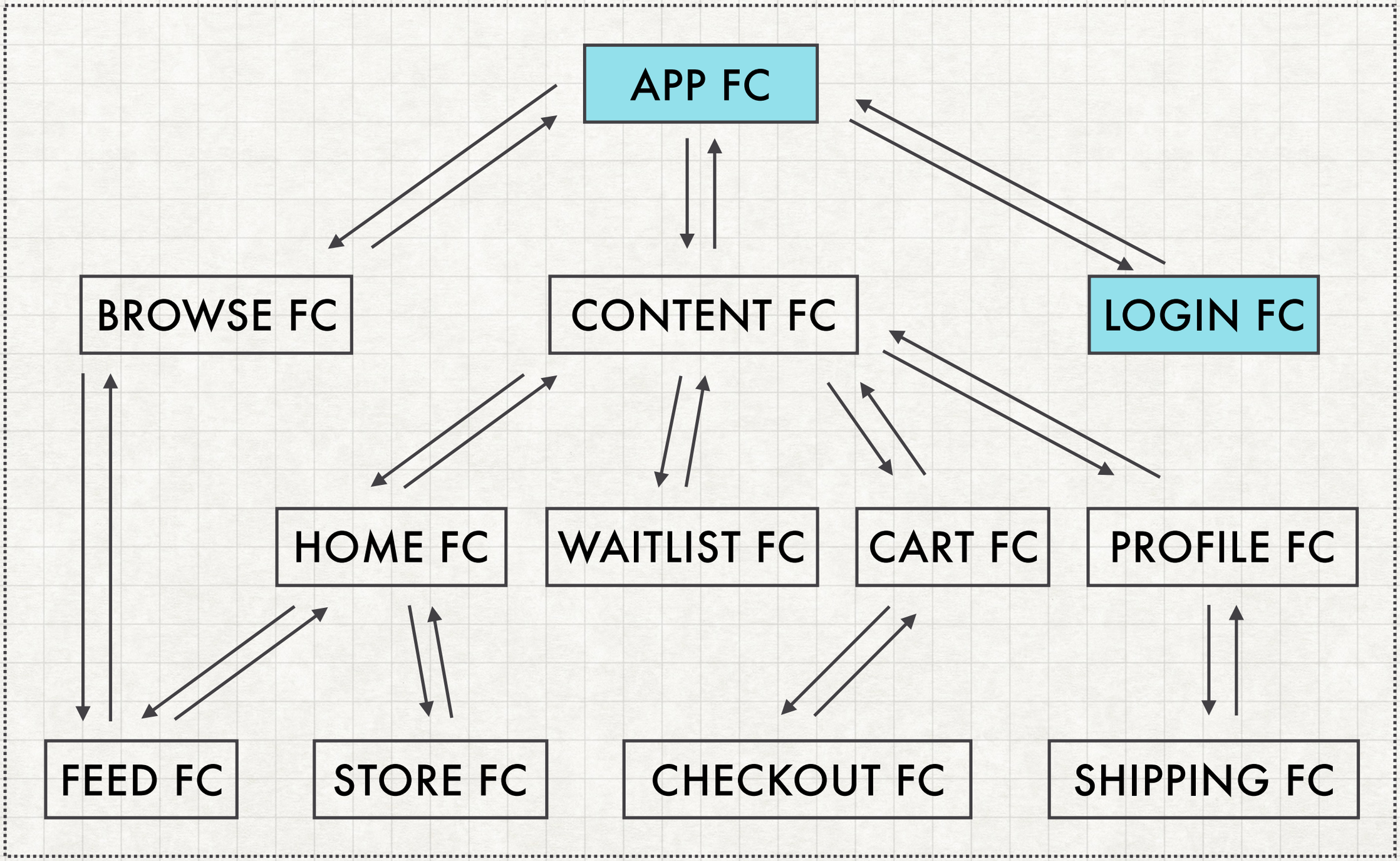
“

NAVIGATION LOGIC IS ABSTRACTED AWAY FROM VIEW CONTROLLERS

”

```swift
// in AppFC...

public func start() {
    autoLogin() { (error) in
        if let error = error {
            self.startLoginFC()
        } else {
            self.startContentFC()
        }
    }
}
```

COMMENT SOLD FLOWS



```
// in AppFC...

private func startLoginFC() {
    let loginFC = LoginFC(withServices: services)
    loginFC.delegate = self
    loginFC.start()
    addChildFC(loginFC)
    rootViewController.present(
      loginFC.rootViewController,
      animated: false,
      completion: nil
    )
}
```

```
// in AppFC...

private func startLoginFC() {
    let loginFC = LoginFC(withServices: services)
    loginFC.delegate = self
    loginFC.start()
    addChildFC(loginFC)
    rootViewController.present(
      loginFC.rootViewController,
      animated: false,
      completion: nil
    )
}
```

“

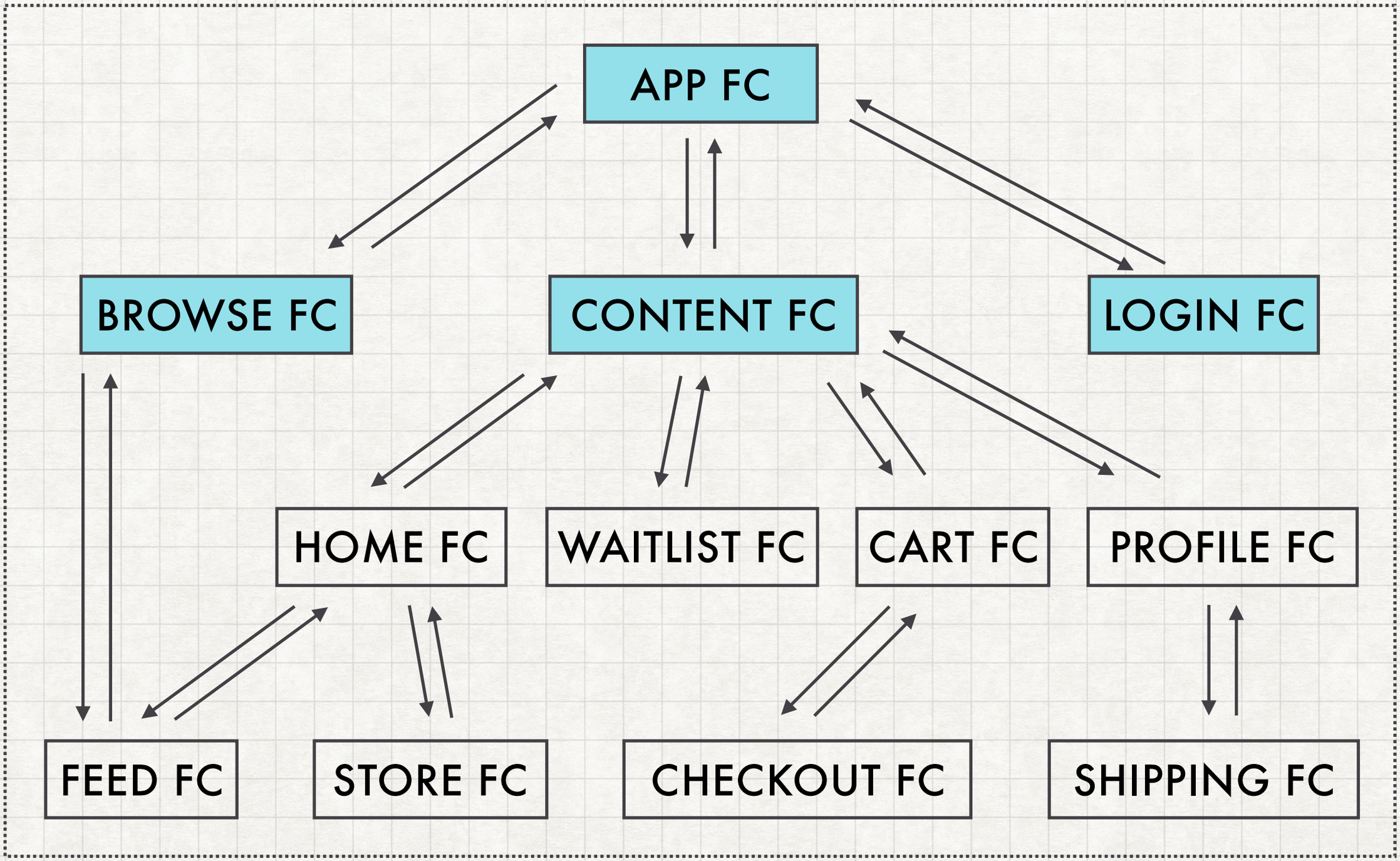
COMMUNICATION IS EXPLICIT AND PROVIDES EASY-TO-USE HOOKS

”

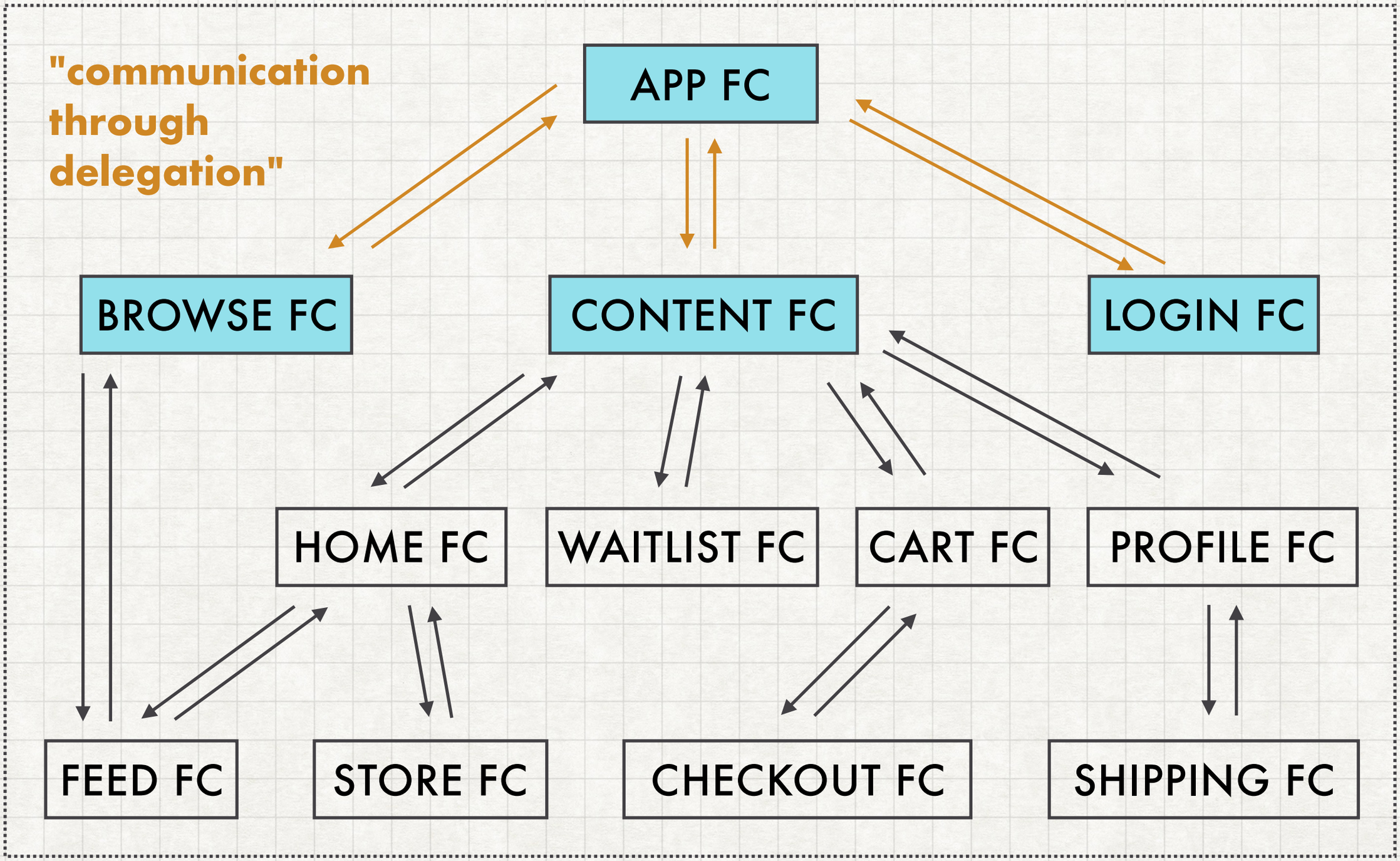
```
// MARK: - LoginFCDelegate

protocol LoginFCDelegate: class {
    func loginFCDidLogin(loginFC: LoginFC)
    func loginFCDidSelectBrowse(loginFC: LoginFC)
}
```

```
// MARK: - AppFC: LoginFCDelegate

extension AppFC: LoginFCDelegate {
    func loginFCDidLogin(loginFC: LoginFC) {
        loginFC.rootViewController.dismiss(animated: false) {
            self.removeChildFC(loginFC)
            self.startContentFC()
        }
    }

    func loginFCDidSelectBrowse(loginFC: LoginFC) {
        loginFC.rootViewController.dismiss(animated: false)
        removeChildFC(loginFC)
        stateBrowseFC()
    }
}
```

COMMENT SOLD FLOWS



COMMENT SOLD FLOWS



“

#DEMO: ADD A NEW FLOW

”

THAT'S IT. THANK YOU!

QUESTIONS?