Carnegie Mellon

08-723
Mobile Application development for iOS and Android

Sophie Saeyoung Jeong

Carnegie Mellon

Outline

- 08-723 Class 101
- Web Service (Continued)
  - Facebook
  - Other Web Services
- REST API Client Design Tips - Concurrency Design
  - NSThread
  - NSOperation
  - GCD (Grand Central Dispatch)

Carnegie Mellon

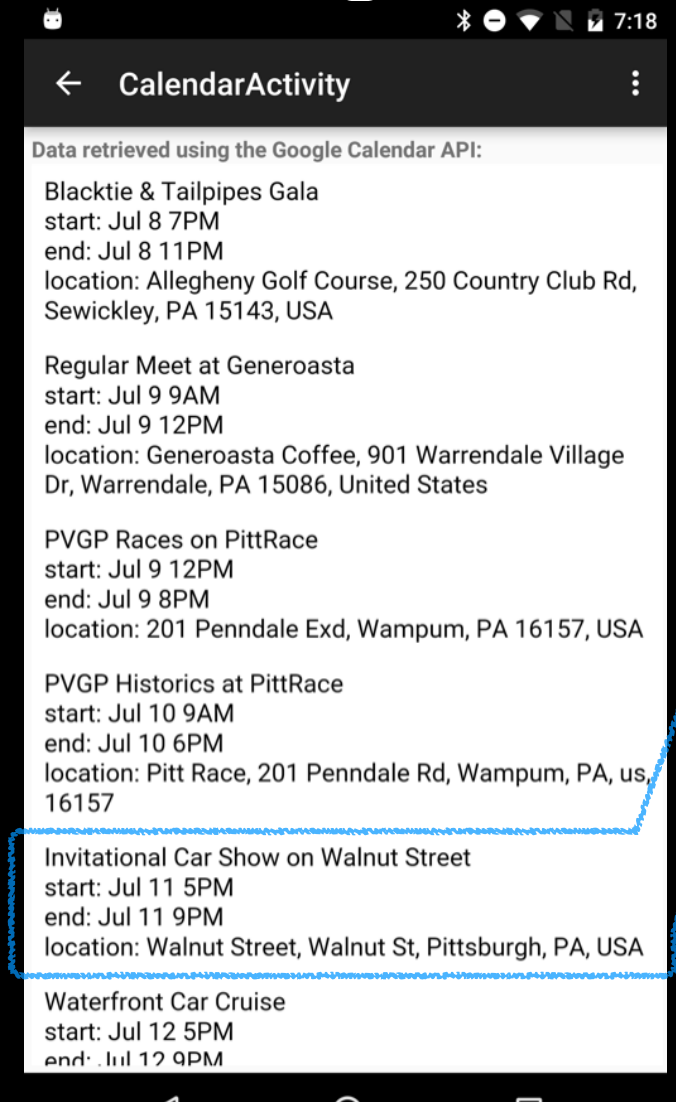
08-723 Class 101
HW7(10%) by July 14th Noon

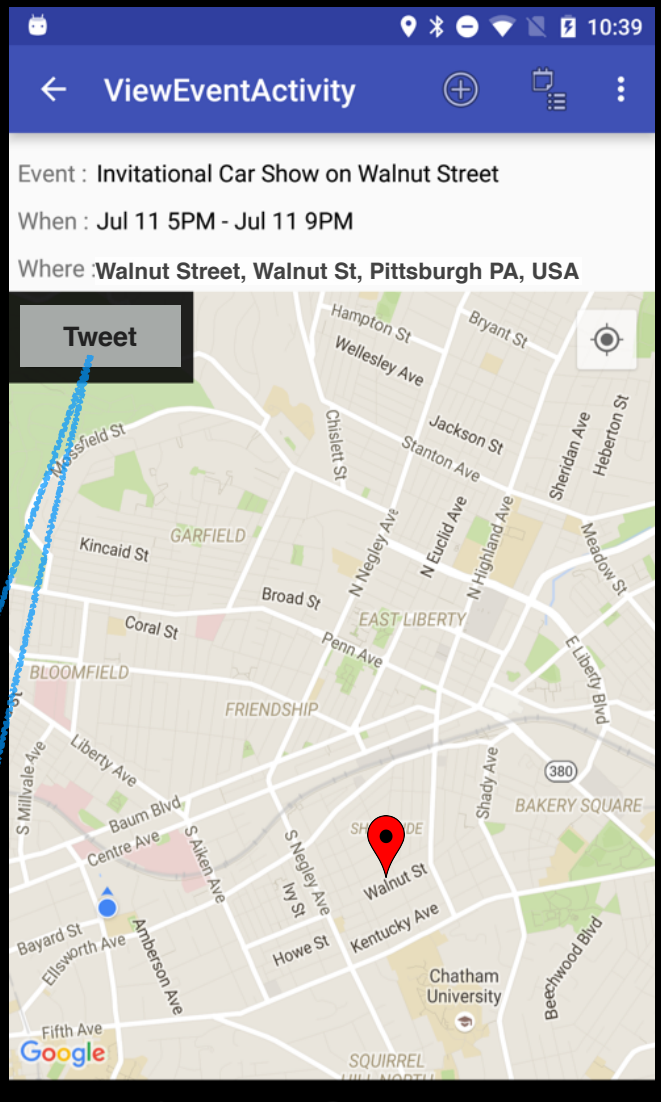
- HW7 due on July 14th Noon - Hardware Deploy Required
  - In the class, Bring your app running on your phone to the class for testing
- Use Google Calendar API
  - Use CalendarID: 5lr4v461mdd2pu55onfcu6prl8@group.calendar.google.com
  - FirstTap: Read calendarEvents in a Table/List View
    - event: Title(summary), start(dateTime), end(dateTime), location
  - View Detail Calendar Event:
    - Display Location info on the Map underneath the detail Event Information (ref. HW4)
    - Add Tweet Button and tweet detailCalendar Info to @MobileApp4 (ref: HW5)
  - Second Tap: Read Tweet your Home TimeLine
    - Use **Home_Timeline** (https://dev.twitter.com/rest/reference/get/statuses/home_timeline)

Carnegie Mellon

08-723 Class 101
HW7: Google Calendar


```
-{
  "kind": "calendar#events",
  "etag": "\"p33sav34kobdsq0g\"",
  "summary": "2014 iDrift",
  "description": "08723 course example - Drifting
  "updated": "2016-07-05T04:46:23.275Z",
  "timeZone": "America/New_York",
  "accessRole": "owner",
   "kind": "calendar#event",
   "etag": "\"2812382340460000\"",
   "id": "6au7q0qrl16f46a0emgtleft9k",
   "status": "confirmed",
   "htmlLink": "https://www.google.com/calen
eid=NmF1N3EwcXJsMTZmNDZhMGVtZ3RsZWZ0OWsgNWxy
   "created": "2014-07-24T08:39:30.000Z",
   "updated": "2014-07-24T08:39:30.230Z",
   "summary": "Downtown Parade",
  +"creator": {
     ...
   },
  +"organizer": {
     ...
   },
  -"start": {
     "dateTime": "2014-07-24T04:39:30-04:00"
   },
  -"end": {
     "dateTime": "2014-07-24T05:39:30-04:00"
   },
```



Carnegie Mellon

08-723 Class 101 Final Project Begins…

- Final Presentation + Live/Youtube Demo (20%)
- Final Report , [iOS + Android] Project Submission (20%)
- Demo - Live Presentation and a YouTube Video (Both)
- Your Final Project Report.....
  - Solution
    - How you feed data from External Sensors and extend with Web Services
    - Architecture : Data Persistency, Asynchronous Mechanism,etc
    - Differences between iOS and Android
  - UX: How you Improved User Experience

Carnegie Mellon

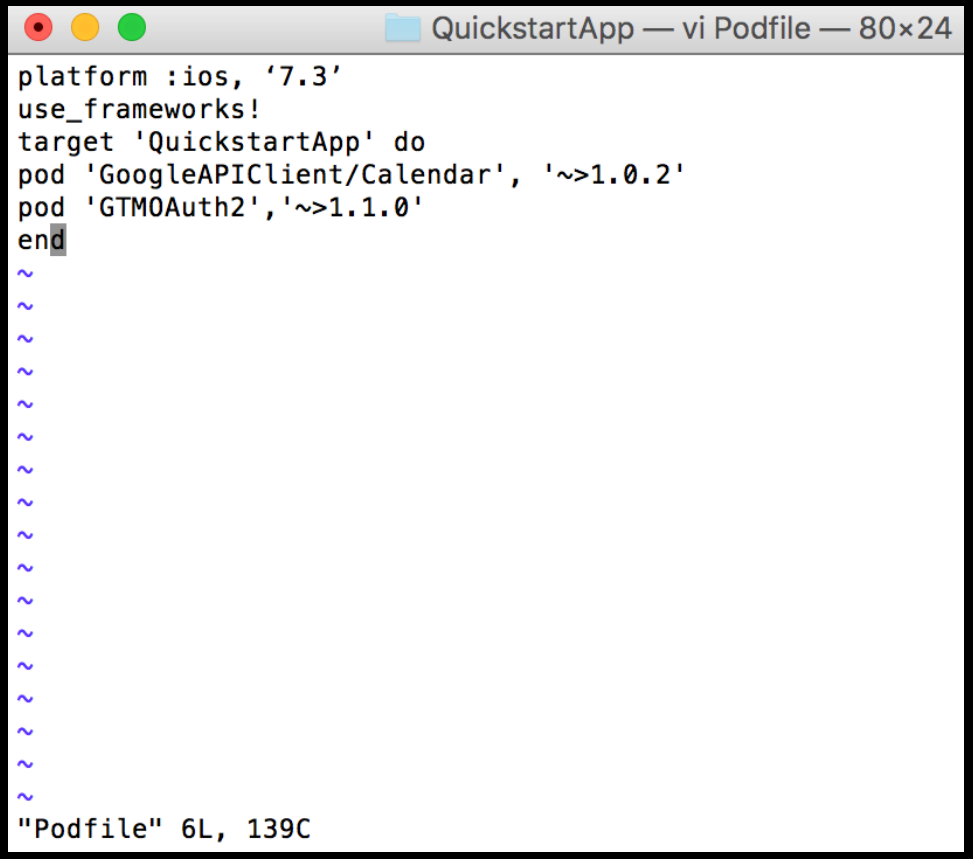
08-723 Class 101
Project Final

- Office Hours: Fridays 12PM-1:30PM at 417 S.Craig #202A
- Final Project Submission Due : Aug. 1st
  - Source Code *.zip format
  - Final Report
  - Youtube Videos : Post Youtube Link on Piazza
  - Presentation File
- HW8 Submission Due : Aug. 4th
  - Course Evaluation

Carnegie Mellon

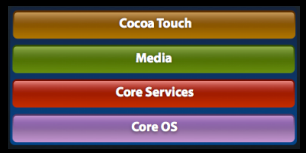
08-723 Class 101 Dependency - CocoaPods

- sudo gem install cocoa pods
- sudo gem install -n /usr/local/bin/ cocoapods
- pod init
- open -a Xcode Podfile
- vi Podfile
- pod install
- open *yourApp*.xcworkspace

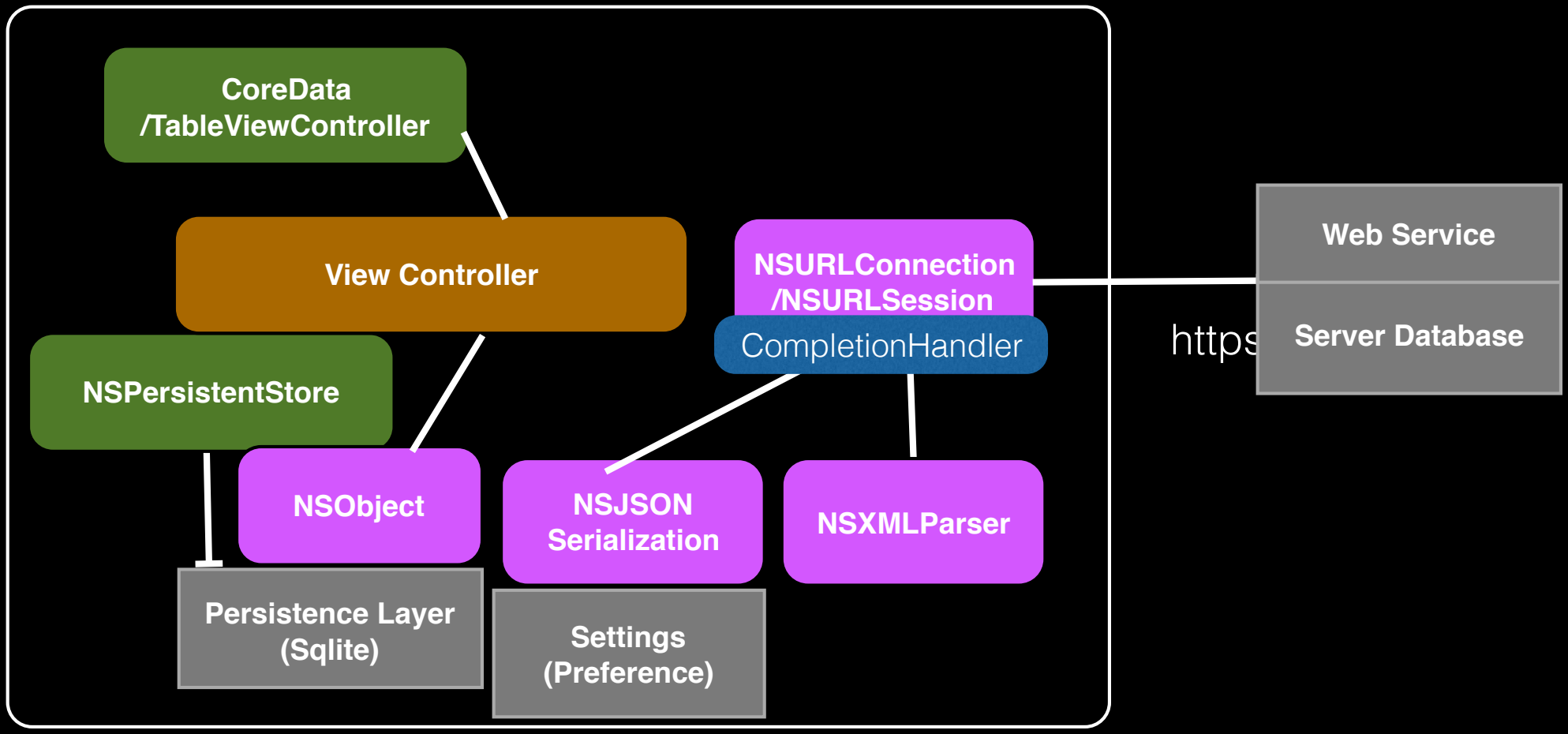


Carnegie Mellon

08-723 Class 101
iOS Client Architecture



iOS



Carnegie Mellon

08-723 Class 101 Build.gradle (Module:app)

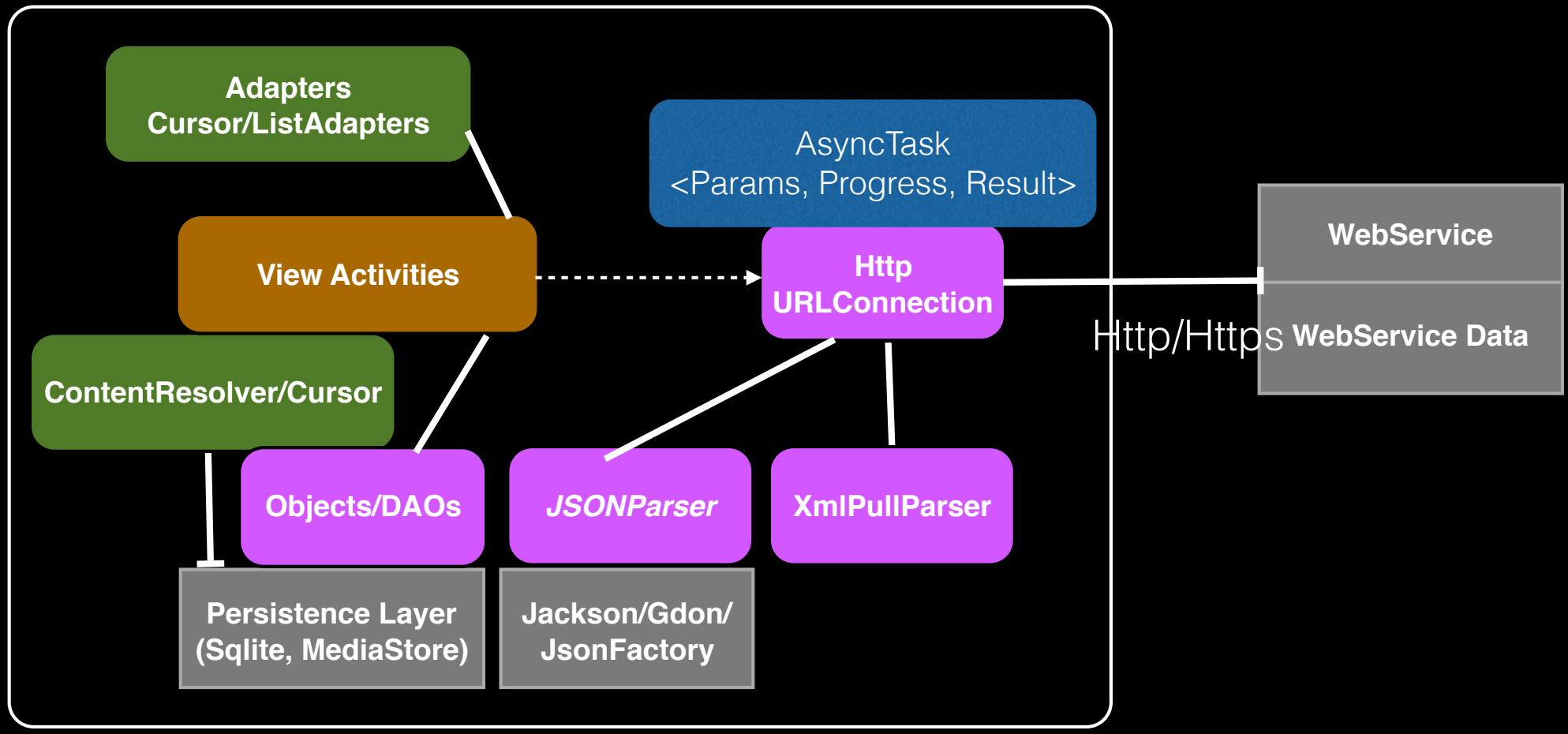
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile files('libs/twitter4j-core-4.0.1.jar')

    compile('com.google.api-client:google-api-client-android:1.22.0') {
        exclude group: 'org.apache.httpcomponents'
    }
    compile('com.google.apis:google-api-services-calendar:v3-rev196-1.22.0
        exclude group: 'org.apache.httpcomponents'
    }


    compile 'com.android.support:appcompat-v7:23.4.0'
    compile 'com.android.support:support-v4:23.4.0'
    compile 'com.google.android.gms:play-services-auth:9.0.2'
    compile 'pub.devrel:easypermissions:0.1.5'
    compile 'com.google.android.gms:play-services:9.0.2'
    compile 'com.google.android.gms:play-services-maps:9.0.2'
    compile 'com.google.android.gms:play-services-location:9.0.2'
    compile 'com.google.api-client:google-api-client-gson:1.22.0'
    compile 'com.android.support:design:23.4.0'
}
```

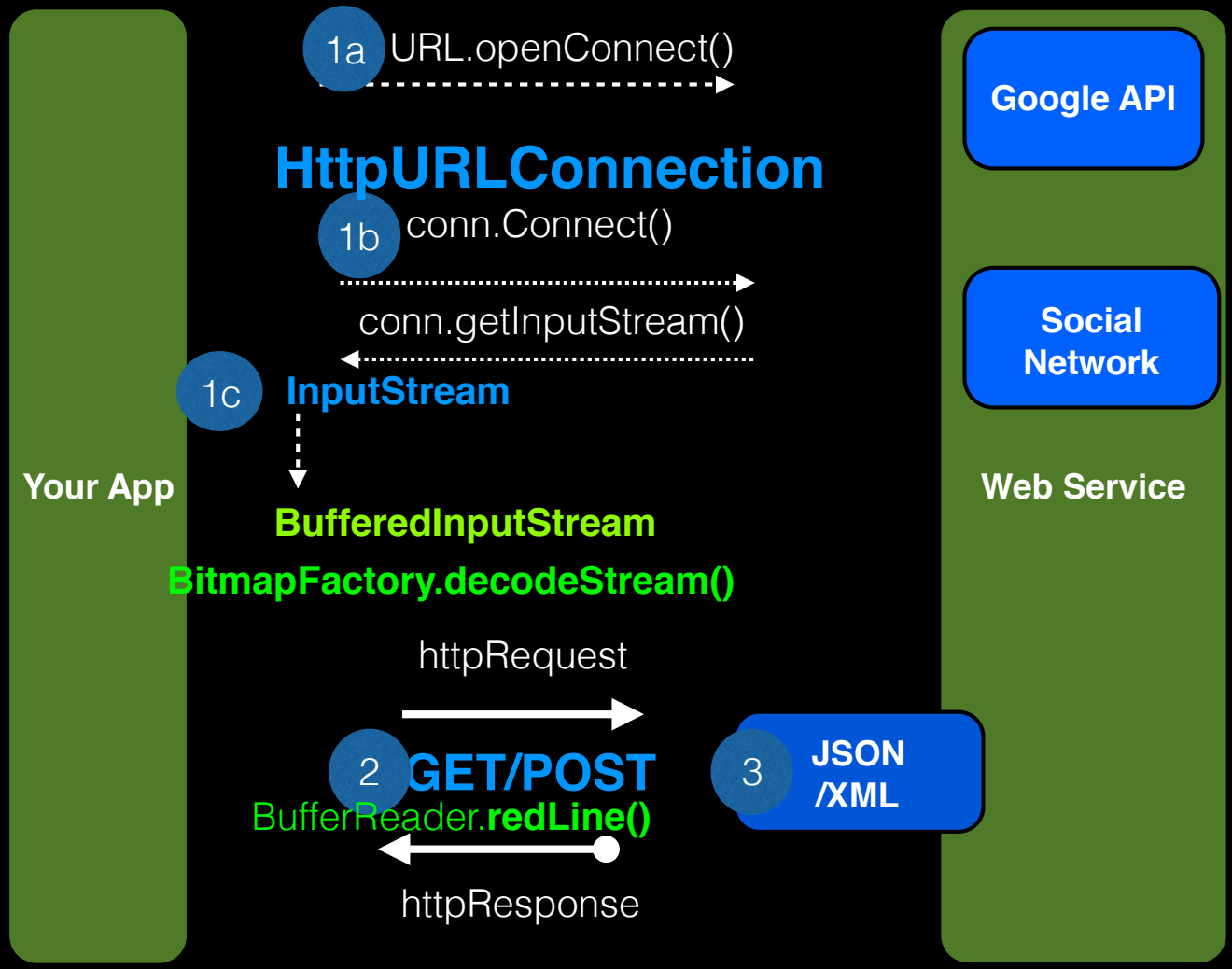
Carnegie Mellon

08-723 Class 101
Android Client Architecture
iOS



Carnegie Mellon

WebServices
Data Handling/Media/Json/XML



Carnegie Mellon

WebService
Other WebService API
iOS

Carnegie Mellon

Facebook
1) iOS Social Framework

- Retrieve Facebook Wall (c.f. Twitter Timeline) - SLRequest
- Call on the ACAccountStore for [ACAcountStore requestAccessToAccountWithType: ACAccountTypeIdentifierFacebook options: completion: ^(BOOL granted, NSError *error)
- set Options
  - BASIC Permission via ACFacebookAudienceKey on ACAccountStore

```
NSDictionary *options = @{
ACFacebookAudienceKey: ACFacebookAudienceEveryone
        ACFacebookAppIdKey: @"yourAppID",
        ACFacebookPermissionsKey: @[@"email"],
};
```

https://developers.facebook.com/docs/

Carnegie Mellon

Facebook
1) iOS Social Framework

- Retrieve Facebook Wall (c.f. Twitter Timeline) - SLRequest
- Publishing a Stream Permission
  - request READ/WRITE permission

```
NSDictionary *options = @{
ACFacebookAudienceKey: ACFacebookAudienceEveryone
        ACFacebookAppIdKey: @"yourAppID",
        ACFacebookPermissionsKey: @[@"publish_stream"],
};
```

- Feed endpoint: https://graph.facebook.com/me/feed
- Photos endpoint: https://graph.facebook.com/me/photos

https://developers.facebook.com/docs/

Carnegie Mellon

2) Facebook SDK - iOS

- Install Facebook iOS SDK and Add Facebook iOS SDK
  - FBSDKCoreKit.Framework
  - FBSDKLoginKit.Framework
  - FBSDKShareKit.Framework
- Provide Bundle ID on the Facebook
- Open yourApp.plist
  - Add **FacebookAppID**,
  - **FacebookDisplayName**
- Add Facebook Login Button and Permissions

```
<key>CFBundleURLTypes</key>
<array>
  <dict>
  <key>CFBundleURLSchemes</key>
  <array>
    <string> yourfacebookAppID </string>
  </array>
  </dict>
</array>
<key>FacebookAppID</key>
<string> yourfacebookAppID </string>
<key>FacebookDisplayName</key>
<string>iDrift</string>
```

https://developers.facebook.com/docs/facebook-login/ios

Carnegie Mellon

Facebook SDK Android

- Configure and Linked your App on the Facebook (you need **FacebookApp ID**) on https://developers.facebook.com/apps
- Import Facebook Android SDK and Add SDK to build.grade
- Provide Package Info
- Add **Android Key Hash** to developer Profile (https://developers.facebook.com/settings/developer/contact/)
- Facebook Login - LoginButton or LoginManager
  - https://developers.facebook.com/docs/facebook-login/android
- Permissions on Android: https://developers.facebook.com/docs/facebook-login/android/permissions

https://developers.facebook.com/docs/facebook-login/android/v2.3

Carnegie Mellon

Facebook SDK Android build.gradle

- Add Facebook Android SDK to project
- Open your App's build.gradle
- Add **Maven Central Repository** to the build.gradle

```
repositories { mavenCentral() }
```

- Add 'compile com.facebook.android:facebook-android-sdk: ' in the dependencies
- Build your project
- Add FacebookSDK: import com.facebook.FacebookSdk;

https://developers.facebook.com/quickstarts/1601708976794351/?platform=android

Carnegie Mellon

Facebook SDK Android AndroidManifest.xml

- open AndroidManifest.xml
- Add a user-permission INTERNET

```
<uses-permission android:name="android.permission.INTERNET"/>
```

- Add a meta-data to the application

```
<application android:label="@string/app_name" ...>
    ...
    <meta-data android:name="com.facebook.sdk.ApplicationId" android:value=
    ...
</application>
```

"yourfacebookAppID"

https://developers.facebook.com/quickstarts/160170897679435 1/?platform=android

Carnegie Mellon

Webservice API
Parse Framework

- www.parse.com

Data

Push

Analytics

Social

Cloud Code

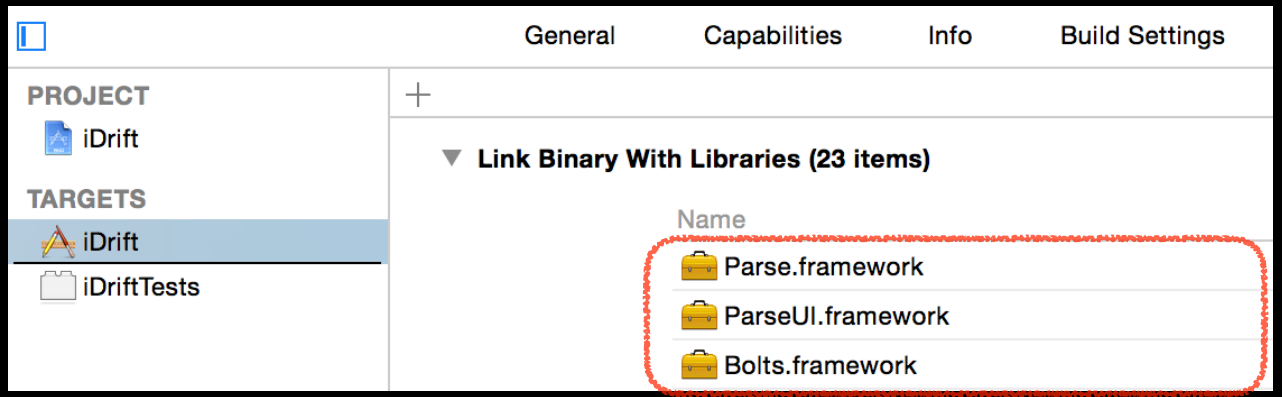
Hosting

Embedded

Carnegie Mellon

Webservice API Parse Framework

- iOS - Static Library Parse SDK, Use Cocoapods
  Pod 'Parse-iOS-SDK'



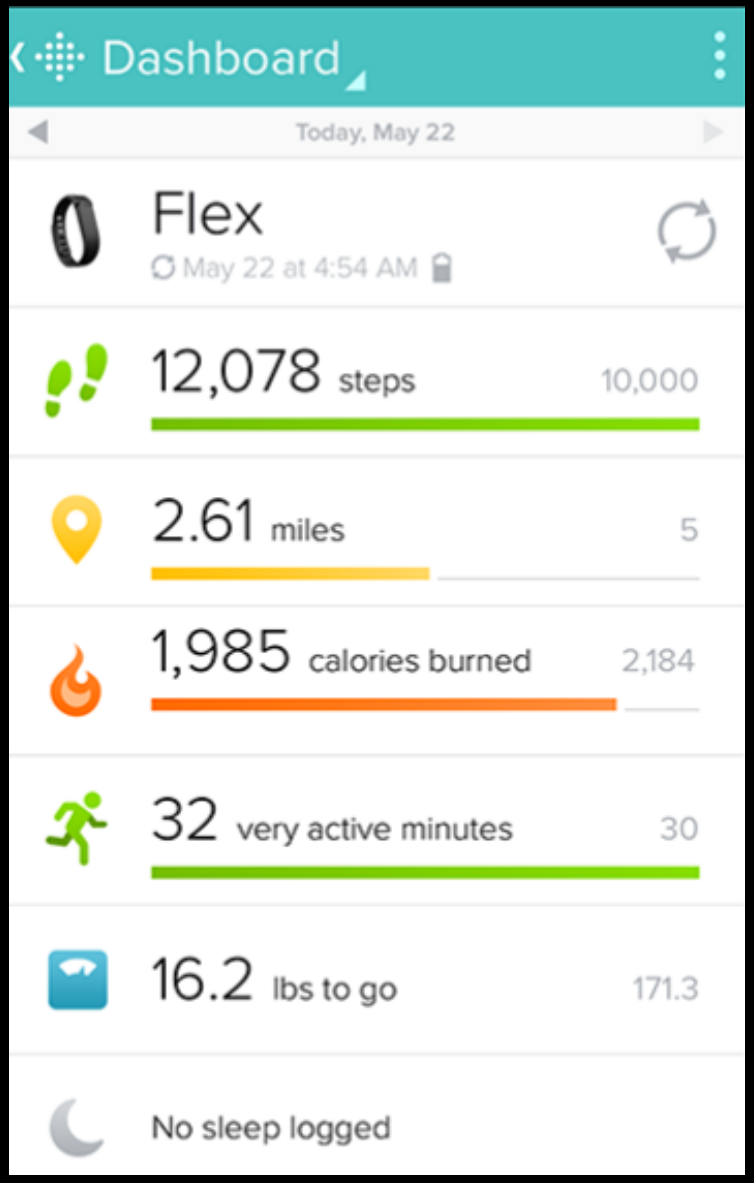
- Android : lib/parse.jar , dependency in gradle

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile files('libs/twitter4j-core-4.0.1.jar')
    compile 'com.parse.bolts:bolts-android:1.+'
    compile fileTree(dir: 'libs', include: 'Parse-*.jar')
```

Carnegie Mellon

Fitbit API

- OAuth Authentication
- Fitbit Resource Access API
- Fitbit Subscription API
- JSON >> XML (~ 5%)
- Rate Limit
- API Time Zone
- OAuth IO: https://oauth.io/home



https://wiki.fitbit.com/display/API/

Carnegie Mellon

Other Web Services API

- Find a restaurant in a map, book a ride with uber, cook, connected Cars etc
  - Messaging: WeChat, Slack , Twillio etc
  - BookingaRide:Uber,Lyft,etc
  - Photo Search: Pinterest (Shoppers), Instagram , Flickr, etc
  - Workout: RunKeeper, Runtastic, Fitbit, WithThings
  - Music: Spotify, SoundCloud, etc
  - Event: Eventful , Google Calendar , etc
  - Car (OBD) : OpenCarData , CarmaLink GPS, Dash Chassis API,etc
  - Food/Restaurant t: Yelp, OpenTable, Nutritionix, Yummly, Webknox API etc

Carnegie Mellon

Concurrency Design

Concurrency with UI Thread

Carnegie Mellon

Concurrency Design
Concurrency with UI thread

- NSThread
- NSOperation
- GCD (grand central dispatch)

Carnegie Mellon

Concurrency Design NSThread

- NSThread
  - Controls its own thread of execution
  - When performing lengthy task, no blocking main UI
  - Performance increase by dividing a large job into smaller jobs
  - Call detachNewThreadSelector:toTarget:withObject:

```
[NSThread detachNewThreadSelector:@selector(exeThread)
toTarget:self withObject:nil];
[NSThread sleepForTimeInterval:1.0]
```

Carnegie Mellon

Concurrency Design NSOperation

- NSOperation<Foundation/Foundation.h>
  - regulates a set of NSOperation
  - Operations within the queue are organized by Priority Level or Dependency
  - Operations can be reused/suspend/cancelled
  - Perform background operation separated from main thread

```
NSOperationQueue *queue = [[NSOperationQueue alloc]init];
```

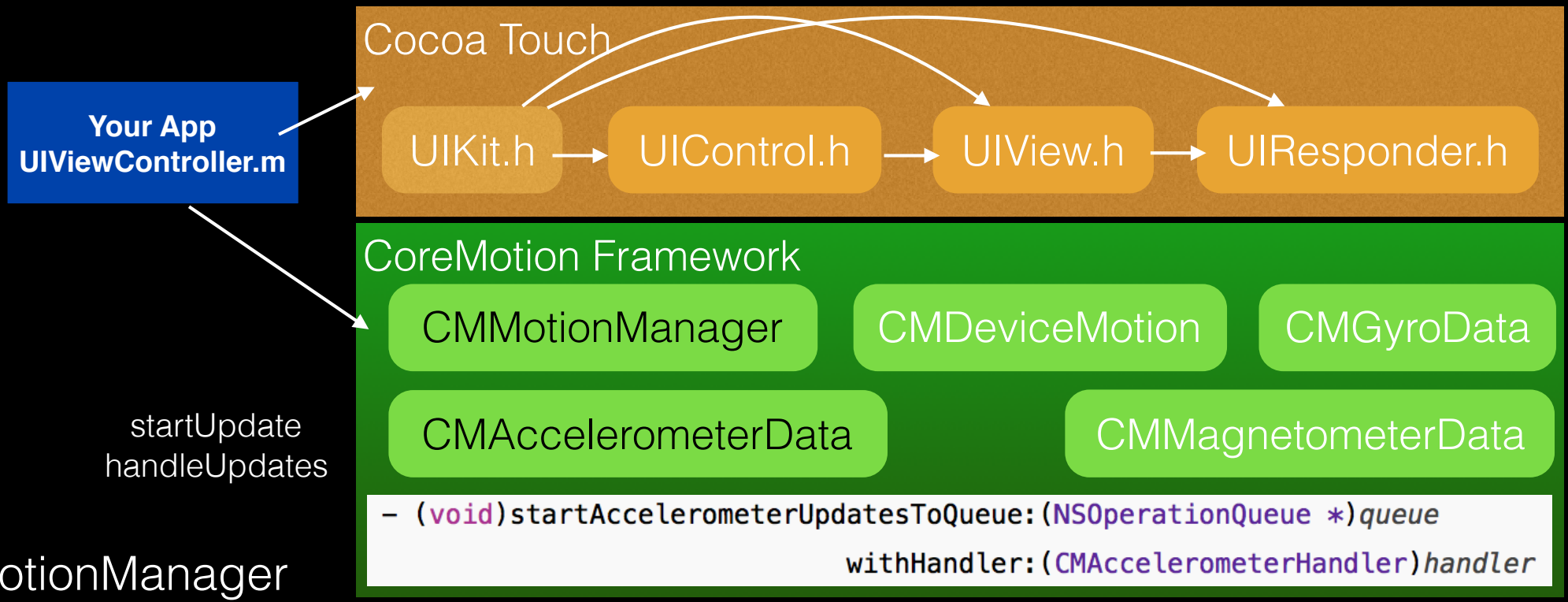
Carnegie Mellon

Concurrency Design NSOperation

- UIOperation
  - Authentication
  - Alerts, Present UI Modally
- BlockOperation
  - NSOperation to execute a block
  - Dependencies for blocks
- URLSessionTask: Adds “conditions”, Adds “observers”

Carnegie Mellon

Concurrency Design
NSOperation - CMFramework




```
[motionManager
StartAccelerometerUpdatesToQueue: [NSOperationQueue queue]
withHandler: ^(CMAccelerometerData *newAccelData , NSErrror *error) {
// handle updates }];
[motionManager stopAccelerometerUpdates];
motionManager = nil;
```

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

- GCD (Grand Central Dispatch) provides queues
  - Creating Queues :

```
dispatch_queue_t dispatch_queue_create( const char *label dispatch_queue_attr_t attr);
```

    - attr: DISPATCH_QUEUE_SERIAL (or NULL), DISPATCH_QUEUE_CONCURRENT
    - e.g.

```
dispatch_queue_t dqueue =
dispatch_queue_create("com.example.myqueue",
NULL);
```

  - Managing Queues:

```
dispatch_queue_t dispatch_get_main_queue(void);
dispatch_queue_t dispatch_get_global_queue
```

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

- Queuing Tasks for Dispatch
  - Queue Types : Main , Concurrent ,Serial
  - Async `dispatch_async( dispatch_queue_t queue, dispatch_block_t block);`
  - Sync `dispatch_sync( dispatch_queue_t queue, dispatch_block_t block);`

```
dispatch_queue_t dqueue =
dispatch_queue_create("com.example.myqueue", NULL);
dispatch_async(dqueue, ^{dispatch_block_t block, result});
dispatch_async(dispatch_get_main_queue(),^{[self do];});
```

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

- Data Type
  - dispatch_queue_t
  - dispatch_block_t
- Constants
  - dispatch_queue_prioroty_t : HIGH, LOW, DEFAULT, BACKGROUND
- Dispatch Queue Type
  - DISPATCH_QUEUE_SERIAL
  - DISPATCH_QUEUE_ CONCURRENT

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

- dispatch_queue_t
  - Serial Queues: maxConcurrentOperationCount=1
  - Concurrent Queues:maxConcurrentOperationCount=Default
- dispatch_block_t
  - Long-running task
  - Subclassable

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

```
@property (nonatomic) dispatch_queue_t sessionQueue;
@property (nonatomic) AVCaptureSession *session;
@property (nonatomic) AVCaptureDevice *videoDevice;
```

```
// Create the AVCaptureSession.
self.session = [[AVCaptureSession alloc] init];

_videoDevice = [AVCaptureDevice
    defaultDeviceWithMediaType:AVMediaTypeVideo];


// Communicate with the session and other session
    objects on this queue.
self.sessionQueue =
    dispatch_queue_create( "session queue",
    DISPATCH_QUEUE_SERIAL );
```

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

```
@property (nonatomic) AVCaptureSession *session;
@property (nonatomic) AVCaptureDevice *videoDevice;
@property (nonatomic) AVCaptureDeviceInput *
    videoDeviceInput;
@property (nonatomic) AVCaptureMetadataOutput *
    metadataOutput;
 _videoDeviceInput = [AVCaptureDeviceInput
     deviceInputWithDevice:_videoDevice error:&
     error];
 if (_videoDeviceInput) {
     [_session addInput:_videoDeviceInput];
 } else {
     NSLog(@"Error: %@", error);
 }

 _metadataOutput = [[AVCaptureMetadataOutput alloc
     ] init];
 [_metadataOutput setMetadataObjectsDelegate:self
     queue:dispatch_get_main_queue()];
 [_session addOutput:_metadataOutput];
 _metadataOutput.metadataObjectTypes =
     [_metadataOutput availableMetadataObjectTypes
     ];
 [_session startRunning];
```

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

```
- (void)geocodeAddressString:(NSString *)addressString
           completionHandler:(CLGeocodeCompletionHandler)completionHandler
typedef void (^CLGeocodeCompletionHandler)(NSArray *placemark, NSError *error);
```

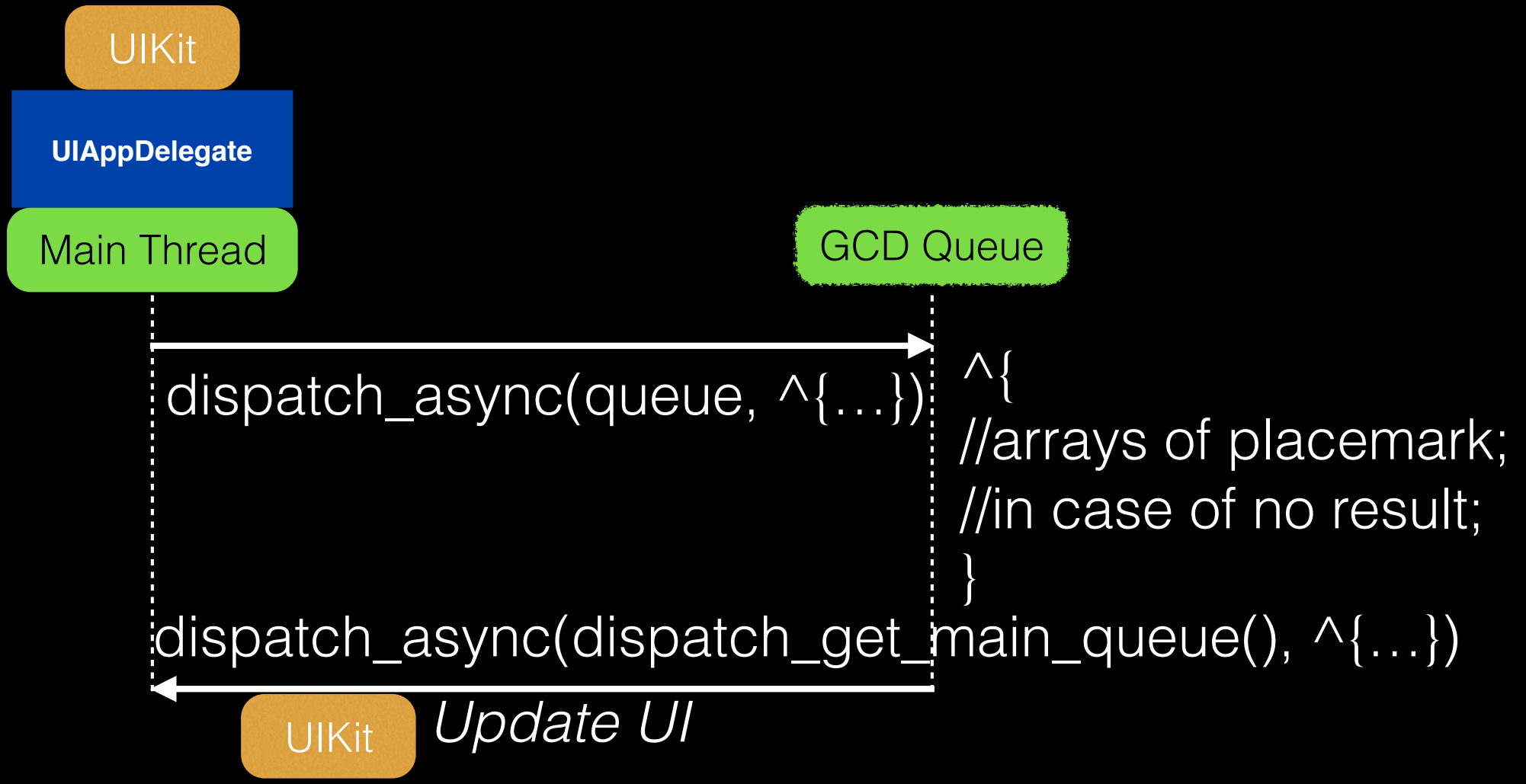
```
// perform geocode
CLGeocoder *geocoder = [[[CLGeocoder alloc] init]
    autorelease];

[geocoder geocodeAddressString:self.
    searchTextField.text completionHandler:^

    dispatch_async(dispatch_get_main_queue(),^ {
```

Carnegie Mellon

Concurrency Design
GCD(Grand Central Dispatch)



Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)

- Posting to Twitter
- Accessing to Twitter Account

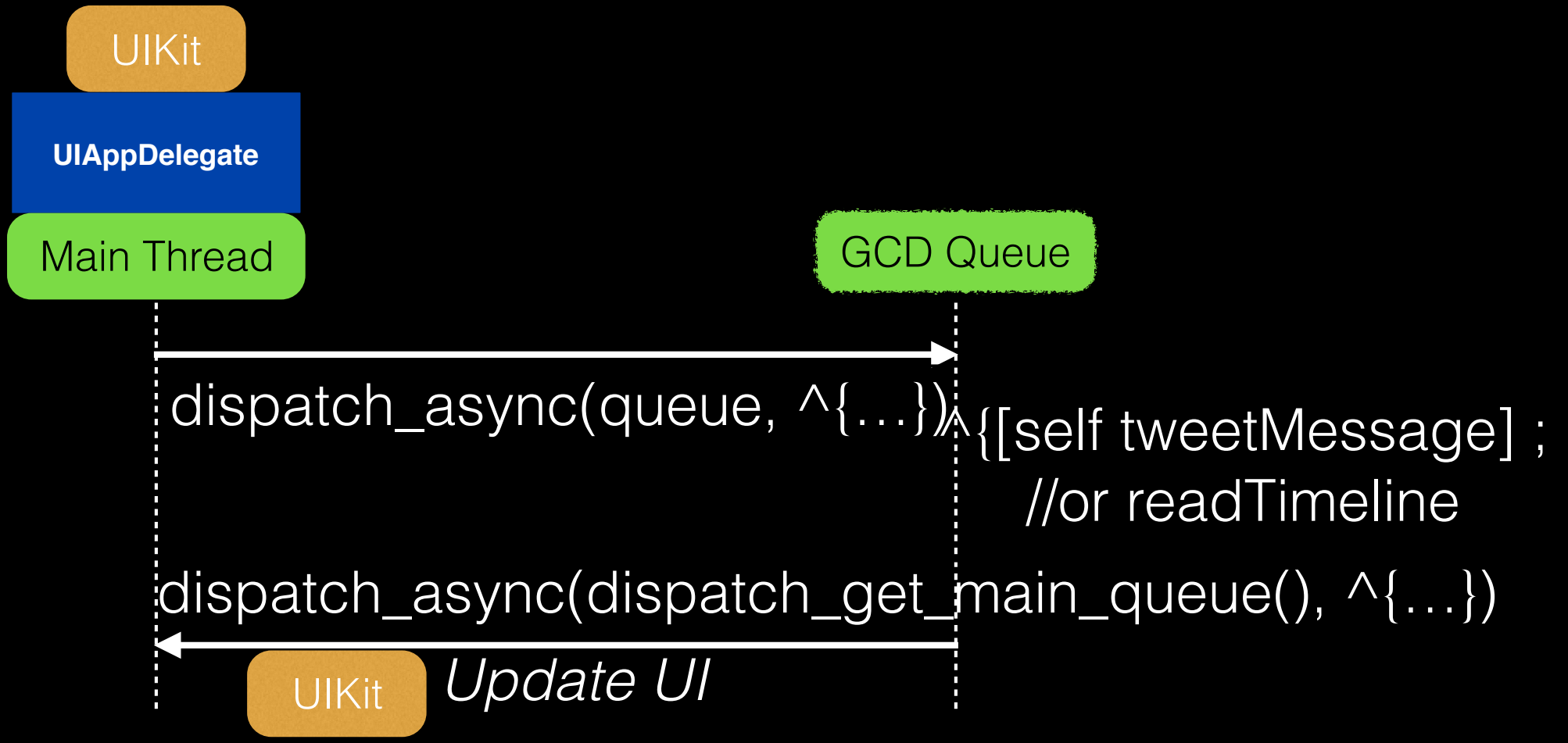
```
- (void)requestAccessToAccountsWithType:(ACAccountType *)accountType
                                options:(NSDictionary *)options
                             completion:(ACAccountStoreRequestAccessCompletionHandler

typedef void(^ACAccountStoreRequestAccessCompletionHandler)(BOOL granted, NSError *error);
```

```
[self.accountStore
    requestAccessToAccountsWithType:twitterAccount
    options:nil completion:^(BOOL granted, NSError
    *error) {
    dispatch_async(dispatch_get_main_queue(), ^{
        [self tweetMessage];                    //
```

Carnegie Mellon

Concurrency Design GCD(Grand Central Dispatch)



Carnegie Mellon

Client Design TIPs
Concurrency with UI thread-Block

- Concurrency with UI thread-Block
  - performSelectorInBackground: withObject:

Non Repetitive Task

  - Use NSBlockOperation with NSOperation Queues
    - With GCD, moving work off the main thread
    - With GCD, schedule Blocks on Dispatch Queue

Large, Repetitive , Async Task: Refreshing Photo Feeds

Carnegie Mellon

Concurrency Design
GCD(Grand Central Dispatch)