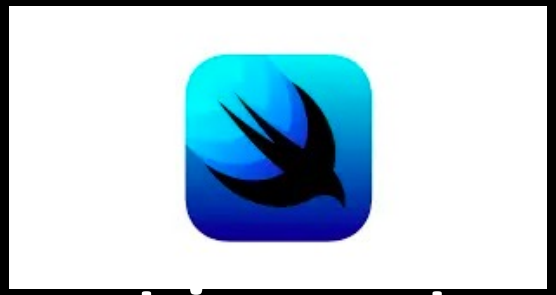


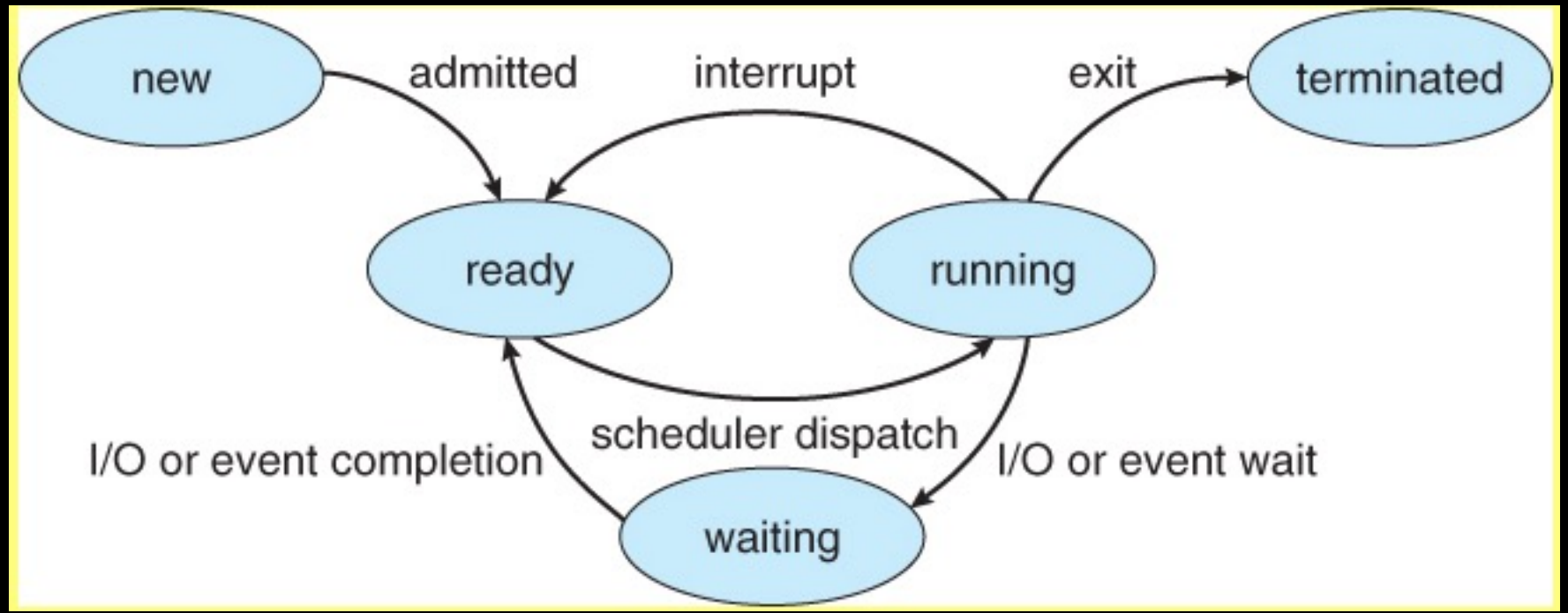
Introduction to SwiftUI Concurrency

Girish Lukka

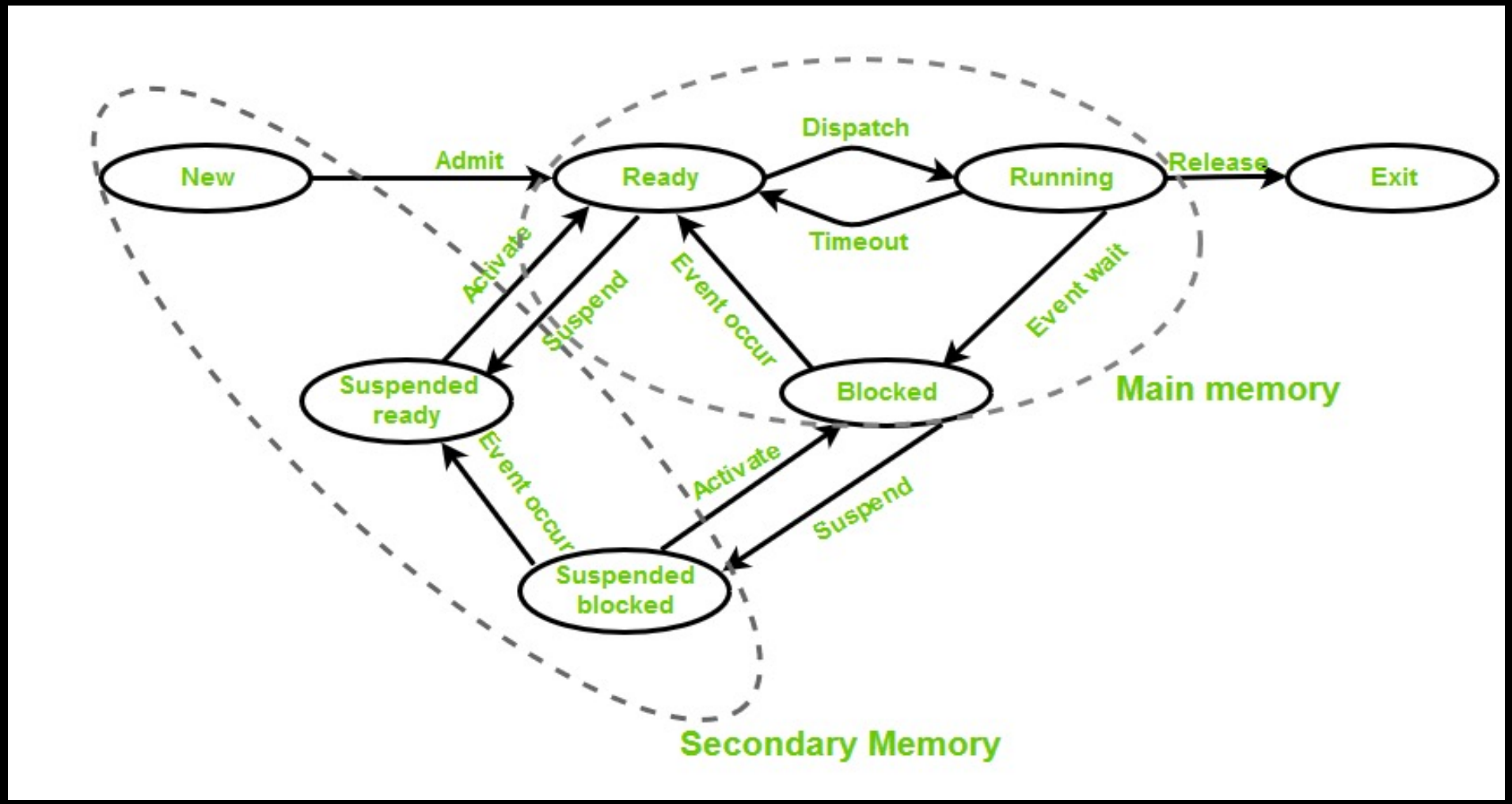
Topics

- **Processes and threads**
- Recap - synchronous and asynchronous using Xcode IDE
- Images – Async and with closures
- Completion handler vs async/await for API calls
- Demos

Computer OS - processes



Computer OS - processes



Processes

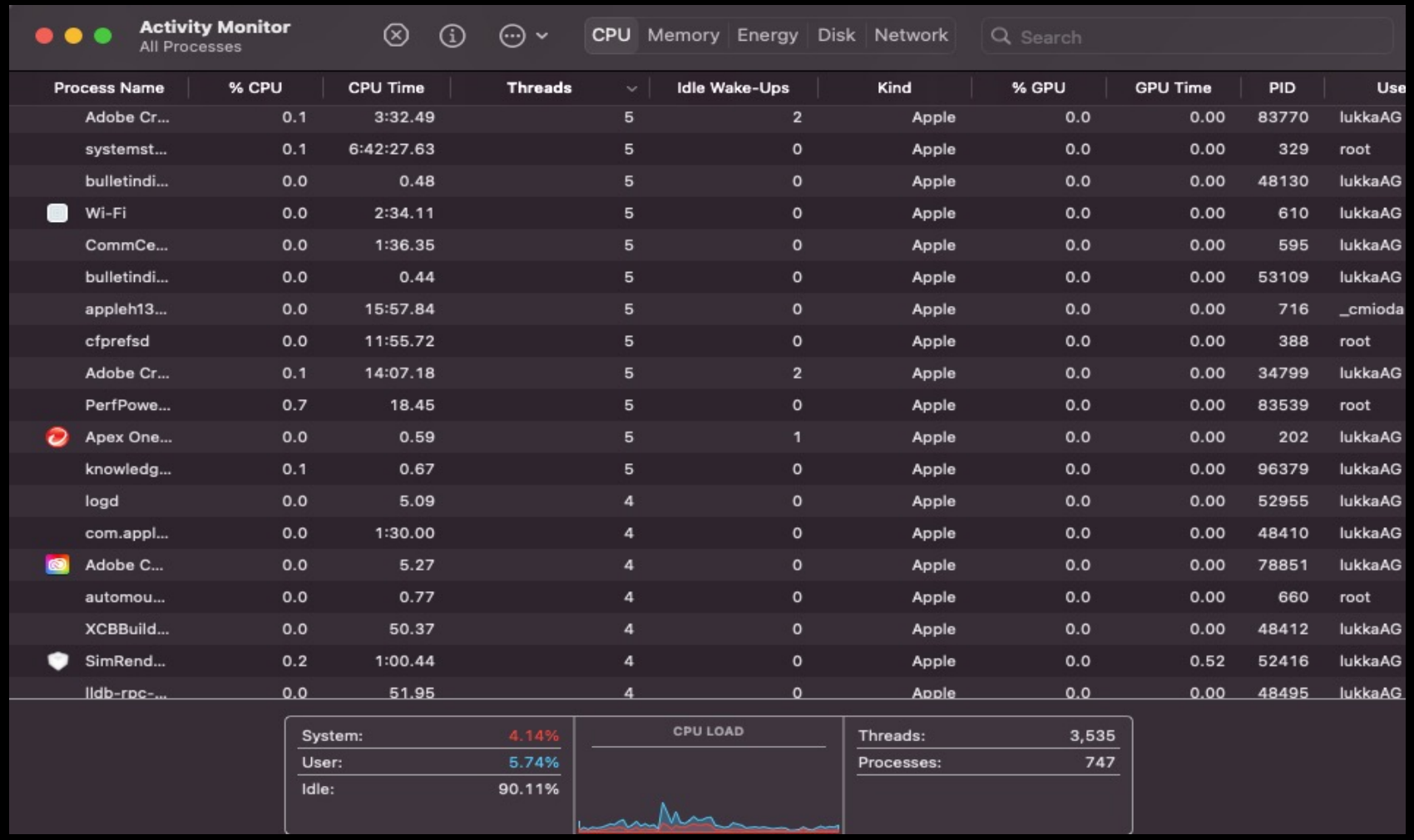
1. Running - the process is currently using the CPU and executing instructions.
2. Ready - the process is waiting to be assigned to a CPU.
3. Blocked - the process is waiting for a resource (such as input/output or a lock) before it can continue.
4. Suspended - the process has been temporarily removed from memory (e.g., swapped out to disk) to free up resources.
5. Terminated - the process has finished executing and has been removed from memory.

- IOS process model is similar with addition of multiple threads an iOS app can have multiple threads of execution, each of which can be in a different state. A thread is a lightweight process that can run concurrently with other threads within the same app.
- Check Task manager in windows threads and handles.

Analogy

- CPA – Critical Path Analysis and Resource allocation
- Critical path analysis is a project planning method that focuses on identifying tasks that are dependant on other tasks for their timely completion.
- Understanding the dependencies between tasks is key to setting a realistic deadline for a complex project.
- Critical path analysis is used in most industries that undertake highly complex projects.
- Optimal resource allocation and avoid bottlenecks

IOS – Activity Monitor-Demo



Recap – Overview API call components

- URL
  - URLRequest
    - URLSession
      - DataTask
        - @State or @ObservedObject

Download data using Data class

```
if let url = URL(string: "https://api.example.com/data.json") {
            do {
                let data = try Data(contentsOf: url)
                // process the downloaded data
            } catch {
                print("Error downloading data: \(error)")
            }
        }
```

- This is a synchronous call so everything will halt until this is completed.

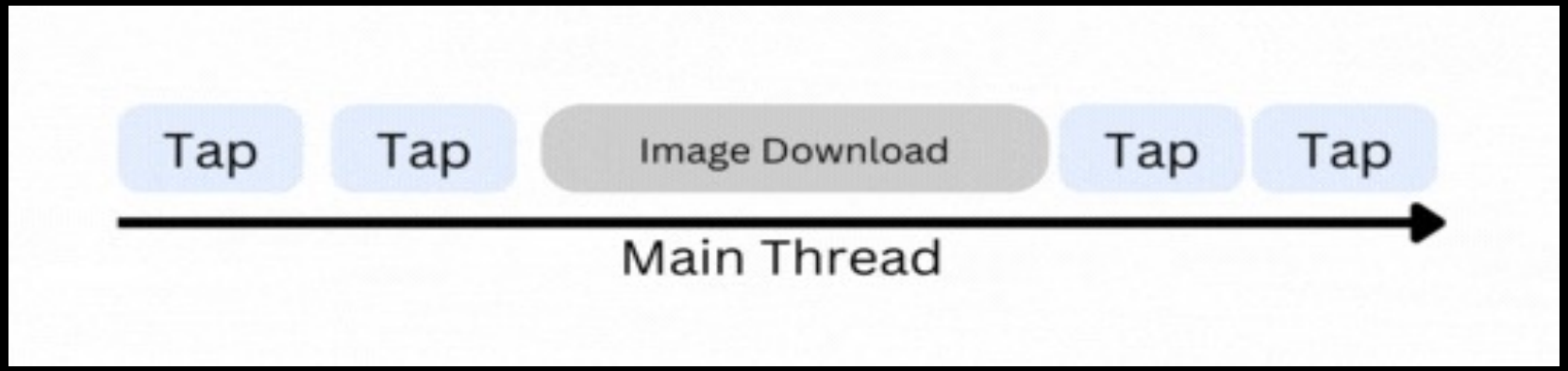
Asynchronous approach

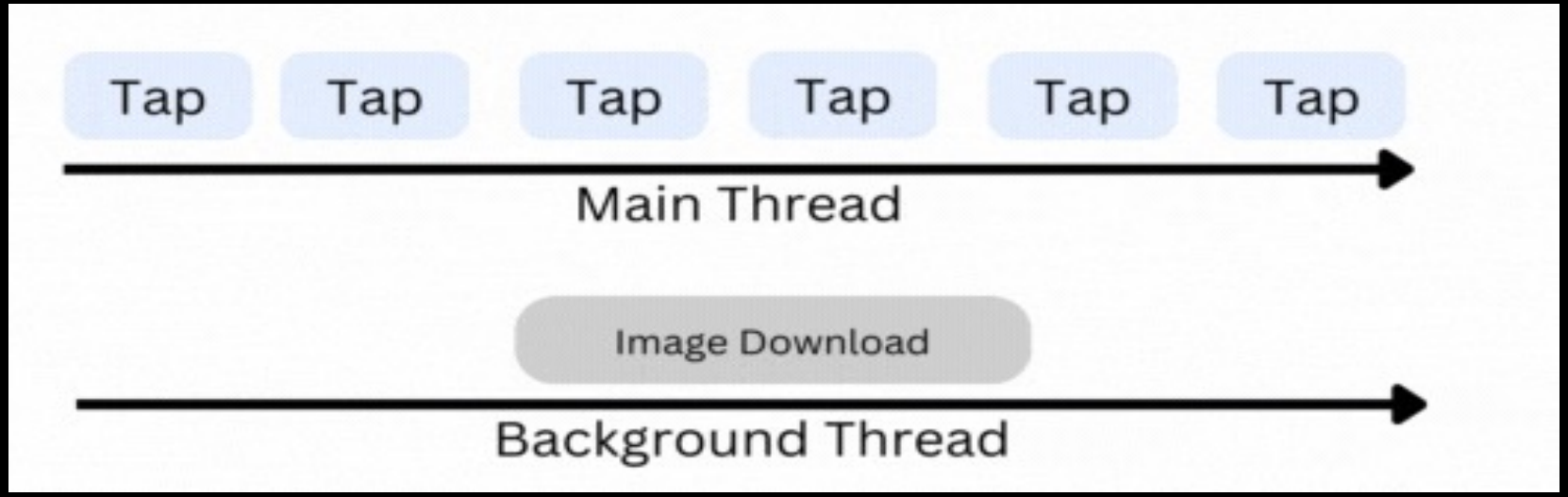
```
func downloadData(from url: URL) async throws -> Data {
    let (data, response) = try await URLSession.shared.data(from: url)
    guard let httpResponse = response as? HTTPURLResponse, httpResponse.statusCode == 200 else {
        throw NSError(domain: "Invalid server response", code: 0, userInfo: nil)
    }
    return data
}

struct ContentView: View {
    @State var data: Data?


    var body: some View {
        VStack {
            if let data = data {
                Text("Downloaded data: \(String(data: data, encoding: .utf8) ?? "")")
            } else {
                Text("No data downloaded")
            }
        }
        .task {
            do {
                let url = URL(string: "https://api.example.com/data.json")!
                self.data = try await downloadData(from: url)
            } catch {
                print("Error downloading data: \(error)")
            }
        }
    }
}
```

Visual representation of Synchronous and Asynchronous tasks

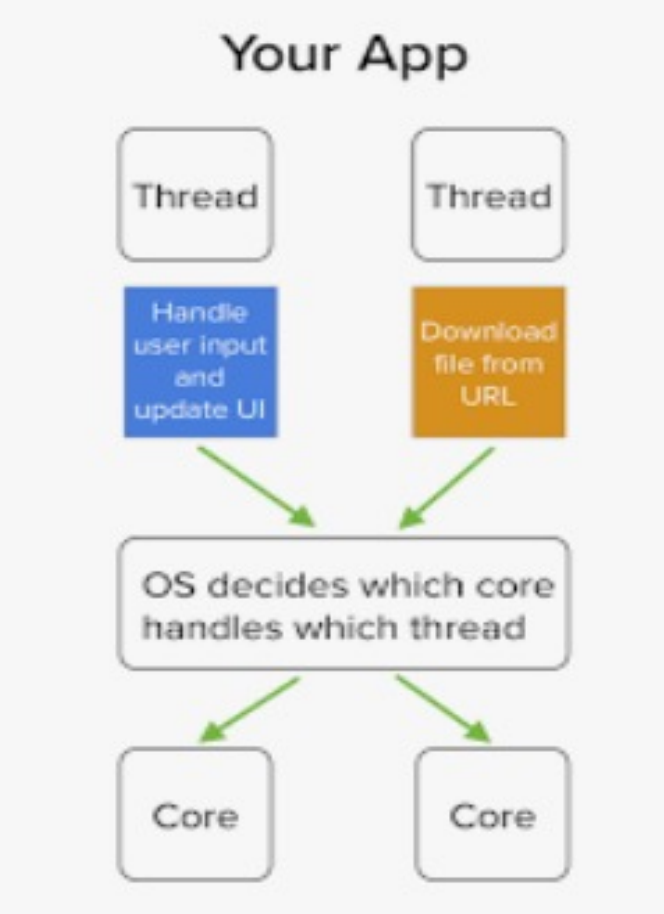


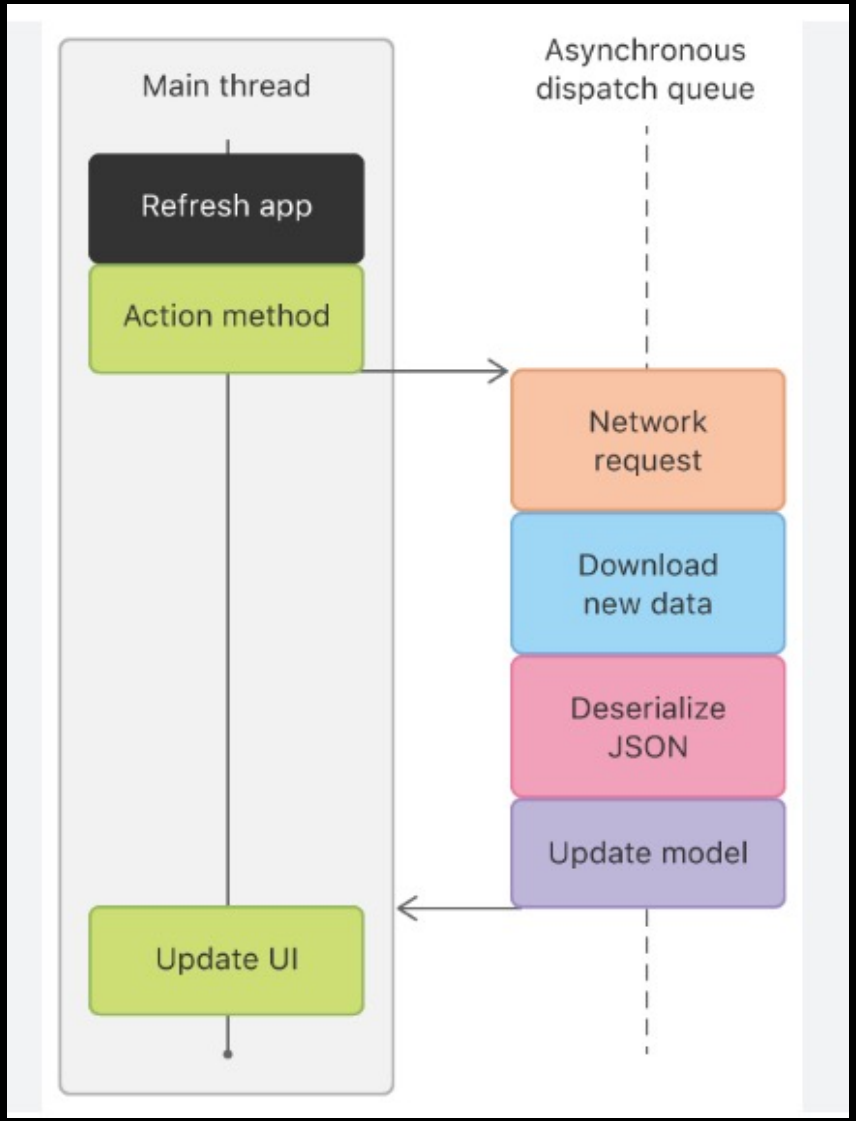


Async and Await

- The main idea behind async await is to offer support for asynchronous task execution without blocking the UI for the user.
- For example, if we have an app that displays an image from the internet and also has a counter, a user can tap on it to increase the counter while waiting for the app.
- Demo 1 – tap and counter

IOS process and thread management





Completion Handler vs Async/Await

- Async URLSession
- Before

```
URLSession
shared
dataTask(with: req) { data, response, error in
// execute something big
}
```

- With async and await:

```
let (data, _) = try await URLSession.shared.data(from: url)
```

- Code demo – coffee app

Resources and additional Reading

- Data Object - Apple Documentation
- Task - Apple Documentation
- URL Loading System - Apple Documentation
- Downloading Data in SwiftUI with URLSession and async/await - Matteo Manferdini
- Network Request using Async/await - LK Seng
- Concurrency - L Kandasamy
- Aysnc/Await – DevTechie