10+ iOS Crashes Every iOS Engineer Must Know (Swift + SwiftUI)

(How to identify them & how to fix them)

Swipe to learn more 

The 10+ iOS Crash Types (At a Glance)

1. Force Unwrap Crash
2. Array / Collection Out of Bounds
3. Dictionary Key Not Found
4. Type Casting Failure (as!)
5. Threading / Main Thread Violation
6. Memory Access Crash
7. Stack Overflow (Infinite Recursion)
8. Fatal Errors & Preconditions
9. SwiftUI State Mutation Crash
10. SwiftUI Object Lifecycle Crash
11. SwiftUI ForEach Crash

1. Force Unwrap(!)

Crash Code: EXC_BAD_INSTRUCTION (SIGILL)

Belongs to:

- Force unwrapping nil optionals
- Unexpected nil values from API / state

Fix it:

- Use guard let, if let
- Provide default values with ??

Old Code (Crash):

```
let name: String? = nil
print(name!) // ❌ CRASH
```

New Code (Fix Crash):

```
if let name = name {
    print(name)
}
```

2. Array / Collection Out of Bounds

Crash Code: EXC_BAD_INSTRUCTION / SIGABRT

Belongs to:

- Accessing invalid index
- Empty arrays
- API data mismatch

Fix it:

- Check bounds
- Use safe APIs like .first

Old Code (Crash):

```
let items = ["A", "B"]
let value = items[2] // ❌ CRASH
```

New Code (Fix Crash):

```
let index = 2
if items.indices.contains(index) {
    let value = items[index]
    print(value)
}
```

3. Dictionary Key Not Found

Title: Dictionary Key Not Found (Forced Access)

Crash Code: EXC_BAD_INSTRUCTION

Belongs to:

- Force unwrapping dictionary values
- Missing or optional keys
- API response mismatch
- Case-sensitive key issues

Fix it:

- Use optional access
- Provide default values
- Validate API keys

Old Code (Crash):

```
let dict = ["name": "Swift"]
let value = dict["age"]!   // ❌ crash
```

New Code (Fix Crash):

```
if let value = dict["age"] {
    print(value)
}
```

4. Type Casting Failure

Crash Code: EXC_BAD_INSTRUCTION (SIGILL)

Belongs to:

- Using as!
- Runtime type mismatch

Fix it:

- Use as?
- Add fallback logic

Old Code (Crash):

```
let value: Any = "10"
let number = value as! Int  // ❌ CRASH
```

New Code (Fix Crash):

```
if let number = value as? Int {
    print(number)
}
```

5. Threading / Main Thread Violation

Crash Code: SIGABRT

Belongs to:

- UI updates outside main thread
- Async background operations

Fix it:

- Use @MainActor
- Dispatch to main queue

Old Code (Crash):

```
DispatchQueue.global().async {
    self.label.text = "Hello"
}
```

New Code (Fix Crash):

```
DispatchQueue.main.async {
    self.label.text = "Hello"
}
```

6. Memory Access Crash

Crash Code: EXC_BAD_ACCESS

Belongs to:

- Accessing deallocated objects
- Weak reference misuse
- Dangling pointers

Fix it:

- Manage object lifecycle
- Use weak self carefully

Old Code (Crash):

```
weak var delegate: MyDelegate?

delegate!.didUpdate()
```

New Code (Fix Crash):

```
delegate?.didUpdate()
```

7. Stack Overflow (Infinite Recursion)

Crash Code: EXC_BAD_ACCESS

Belongs to:

- Infinite recursive calls
- Missing exit condition

Fix it:

- Add termination condition

Old Code (Crash):

```
func fetch() {
    fetch()
}
```

New Code (Fix Crash):

```
func fetch(count: Int) {
    guard count > 0 else { return }
    fetch(count: count - 1)
}
```

8. Fatal Errors & Preconditions

Crash Code: SIGABRT

Belongs to:

- `fatalError()`
- `preconditionFailure()`
- `assertionFailure()`

Fix it:

- Avoid in production
- Handle errors gracefully

Old Code (Crash):

fatalError("Unexpected state")

New Code (Fix Crash):

assertionFailure("Unexpected state")

Return

Rule:

✓ Dev only

❌ Production unsafe

9. SwiftUI State Mutation Crash

Title: SwiftUI State Mutation During View Update

Crash Code: SIGABRT

Belongs to:

- Modifying @State inside body
- Invalid SwiftUI lifecycle usage

Fix it:

- Move state updates to lifecycle events

Old Code (Crash):

```
var body: some View {
    count += 1
    return Text("\(count)")
}
```

New Code (Fix Crash):

```
var body: some View {
    Text("\(count)")
        .onAppear {
            count += 1
        }
}
```

10. SwiftUI Object Lifecycle Crash

Title: SwiftUI ViewModel Deallocation

Crash Code: SIGABRT

Belongs to:

- Incorrect ViewModel ownership
- Using @ObservedObject instead of @StateObject

Fix it:

- Use @StateObject for view-owned objects

Old Code (Crash):

```
@ObservedObject var viewModel = ViewModel()
```

New Code (Fix Crash):

```
@StateObject var viewModel = ViewModel()
```

11. SwiftUI ForEach Crash

Title: SwiftUI ForEach Invalid / Missing Identifier

Crash Code: SIGABRT

Belongs to:

- Non-unique id values
- Using indices that go out of sync
- Using id: \.self on mutable data
- Deleting items without stable identity

Fix it:

- Always provide a **stable, unique ID**

Old Code (Crash):

```
struct Item: Identifiable {
    let id: UUID
    let name: String
}
ForEach(items) { item in
    Text(item.name)
}
```

New Code (Fix Crash):

```
ForEach(items, id: \.id) { item in
    Text(item.name)
}
```

Where to Find Crash Reports

1 Xcode (Development)

- **Xcode → Window → Organizer → Crashes**
- Symbolicated automatically

2 Device Logs

- Connect device
- **Window → Devices & Simulators**
- View logs manually

3 App Store Connect (Production)

- **App Store Connect → TestFlight / App → Crashes**
- Aggregated by OS & device

4 Crashlytics / Third-Party

Most recommended in production:

- Firebase Crashlytics
- Sentry
- Bugsnag

✅ Gives:

- Stack trace
- Device info
- User actions (breadcrumbs)

How Swift Helps Reduce Crashes

Swift provides:

- Optionals
- Strong typing
- ARC memory management
- Compile-time safety
- @MainActor concurrency safety

👉 Most crashes exist **only when we bypass Swift's safety**.

🔑 How to Symbolicate Crash Logs

Requirements:

- .dSYM file
- Crash log
- Matching app build

🔥 Pro Tip

1. **EXC_BAD_ACCESS** = Memory
2. **SIGABRT** = Rule Violation
3. **EXC_BAD_INSTRUCTION** = Unsafe Code

Thank You