MVP Barebones Troubleshooting Guide

For your 4-file CodeKata MVP, here are the most common issues and quick fixes:

🚨 Immediate Build Issues (First 10 Minutes)

Issue: "Cannot find type 'Challenge' in scope"

Symptoms:

```swift
List(challenges) { challenge in // ❌ Cannot find 'Challenge'
```

Root Cause: Files not added to Xcode target

Quick Fix:

1. Select `Challenge.swift` in Xcode navigator
2. Check **File Inspector** (right panel)
3. Ensure your app target is checked under **Target Membership**

```swift
// ✅ Make sure this appears at top of Challenge.swift
import Foundation

struct Challenge: Identifiable {
    // Your code here
}
```

Issue: SwiftUI Preview Crashes

Symptoms:

```
PreviewProvider crashed
Cannot preview in this file
```

Root Cause: Sample data access issues in previews

Quick Fix:

```swift
```

```
// ❌ Problematic preview
#Preview {
    ChallengeDetailView(challenge: Challenge.sampleData[0]) // Crashes if empty
}

// ✅ Safe preview
#Preview {
    if let sampleChallenge = Challenge.sampleData.first {
        ChallengeDetailView(challenge: sampleChallenge)
    } else {
        Text("No sample data available")
    }
}

// ✅ Even safer - create inline sample
#Preview {
    ChallengeDetailView(
        challenge: Challenge(
            title: "Preview Challenge",
            description: "This is a preview",
            difficulty: "Easy"
        )
    )
}
```

Issue: Navigation Doesn't Work

Symptoms:

- Tapping list items does nothing
- Navigation links appear but don't navigate

Root Cause: Missing NavigationView wrapper

Quick Fix:

swift

```swift
// ❌ Navigation won't work
struct ContentView: View {
    var body: some View {
        List(challenges) { challenge in
            NavigationLink(destination: ChallengeDetailView(challenge: challenge)) {
                // Content
            }
        }
    }
}

// ✅ Proper navigation setup
struct ContentView: View {
    var body: some View {
        NavigationView { // Must wrap in NavigationView
            List(challenges) { challenge in
                NavigationLink(destination: ChallengeDetailView(challenge: challenge)) {
                    // Content
                }
            }
            .navigationTitle("CodeKata") // Optional but recommended
        }
    }
}
```

📱 Display & UI Issues (First Hour)

Issue: Text Editor Not Working on Device

Symptoms:

- TextEditor works in simulator
- Real device: keyboard doesn't appear or text doesn't update

Root Cause: iOS keyboard handling differences

Quick Fix:

```swift
// ❌ Basic TextEditor can be problematic
TextEditor(text: $userCode)

// ✅ More robust text editor
TextEditor(text: $userCode)
    .font(.system(.body, design: .monospaced))
    .autocapitalization(.none) // Important for code
    .disableAutocorrection(true) // Important for code
    .border(Color.gray.opacity(0.3))
    .frame(minHeight: 200)
    .onTapGesture {
        // Ensure text editor becomes first responder
    }
```

Issue: List Items Too Small/Large

Symptoms:

- Challenge descriptions cut off
- List rows inconsistent height

- Text overlapping

Root Cause: Missing layout constraints

Quick Fix:

```swift
// ❌ Unpredictable layout
VStack(alignment: .leading) {
    Text(challenge.title)
    Text(challenge.description)
}

// ✅ Controlled layout
VStack(alignment: .leading, spacing: 4) {
    Text(challenge.title)
        .font(.headline)
        .lineLimit(1) // Prevent overflow
    Text(challenge.description)
        .font(.caption)
        .foregroundColor(.secondary)
        .lineLimit(2) // Consistent height
        .fixedSize(horizontal: false, vertical: true) // Allow text wrapping
}
.padding(.vertical, 2) // Consistent spacing
```

Issue: Alert Not Showing

Symptoms:

- Submit button works but no alert appears
- Alert shows but immediately disappears

Root Cause: State management with alerts

Quick Fix:

```swift
```

```swift
// ❌ Common alert mistake
struct ChallengeDetailView: View {
    @State private var showingAlert = false // Correct

    var body: some View {
        VStack {
            Button("Submit Solution") {
                showingAlert = true
            }
        }
        .alert("Success!", isPresented: $showingAlert) { // This might not work
            Button("OK") { }
        }
    }
}

// ✅ Reliable alert pattern
struct ChallengeDetailView: View {
    @State private var showingAlert = false
    @State private var alertTitle = ""
    @State private var alertMessage = ""

    var body: some View {
        VStack {
            Button("Submit Solution") {
                submitSolution()
            }
        }
        .alert(alertTitle, isPresented: $showingAlert) {
            Button("OK") {
                showingAlert = false // Explicit dismissal
            }
        } message: {
```

```
            Text(alertMessage)
        }
    }

    private func submitSolution() {
        alertTitle = "Solution Submitted!"
        alertMessage = "Great job! Your solution has been submitted."
        showingAlert = true
    }
}
```

💾 Data & State Issues (First Day)

Issue: Sample Data Appears Multiple Times

Symptoms:

- Each app launch adds more sample challenges
- List keeps growing
- Duplicate challenges with same content

Root Cause: No persistence means data resets, but adding logic runs every time

Quick Fix:

swift

```swift
// ❌ Adds samples every time
struct ContentView: View {
    @State private var challenges: [Challenge] = Challenge.sampleData

    var body: some View {
        // Every view creation adds samples again
    }
}

// ✅ Initialize once
struct ContentView: View {
    @State private var challenges: [Challenge] = []

    var body: some View {
        NavigationView {
            List(challenges) { challenge in
                // Your list content
            }
            .onAppear {
                loadInitialData()
            }
        }
    }

    private func loadInitialData() {
        if challenges.isEmpty {
            challenges = Challenge.sampleData
        }
    }
}
```

Issue: Changes Don't Persist

Symptoms:

- Complete a challenge, navigate back, progress lost
- App restarts, all changes gone
- User expects data to save

Root Cause: No persistence layer in barebones version

Quick Fix Options:

Option 1: UserDefaults (Simple)

swift

```swift
// ✅ Basic persistence with UserDefaults
class ChallengeStore: ObservableObject {
    @Published var challenges: [Challenge] = []

    private let userDefaults = UserDefaults.standard
    private let challengesKey = "SavedChallenges"

    init() {
        loadChallenges()
    }

    func loadChallenges() {
        if let data = userDefaults.data(forKey: challengesKey),
           let decoded = try? JSONDecoder().decode([Challenge].self, from: data) {
            challenges = decoded
        } else {
            challenges = Challenge.sampleData
        }
    }

    func saveChallenges() {
        if let encoded = try? JSONEncoder().encode(challenges) {
            userDefaults.set(encoded, forKey: challengesKey)
        }
    }

    func markChallengeCompleted(_ challengeId: UUID) {
        if let index = challenges.firstIndex(where: { $0.id == challengeId }) {
            challenges[index].isCompleted = true
            saveChallenges()
        }
    }
}
```

```swift
// Update Challenge struct to support persistence
struct Challenge: Identifiable, Codable {
    let id = UUID()
    let title: String
    let description: String
    let difficulty: String
    var isCompleted: Bool = false // Add completion tracking

    // Custom coding keys for JSON serialization
    private enum CodingKeys: String, CodingKey {
        case title, description, difficulty, isCompleted
    }
}
```

Option 2: In-Memory State Management (Even Simpler)

swift

```swift
// ✅ Shared state without persistence
class AppState: ObservableObject {
    @Published var challenges: [Challenge] = Challenge.sampleData
    @Published var completedChallenges: Set<UUID> = []

    func completeChallenge(_ challengeId: UUID) {
        completedChallenges.insert(challengeId)
    }

    func isCompleted(_ challengeId: UUID) -> Bool {
        completedChallenges.contains(challengeId)
    }
}

// Use in your main app
@main
struct CodeKataApp: App {
    @StateObject private var appState = AppState()

    var body: some Scene {
        WindowGroup {
            ContentView()
                .environmentObject(appState)
        }
    }
}

// Access in views
struct ContentView: View {
    @EnvironmentObject var appState: AppState

    var body: some View {
        List(appState.challenges) { challenge in
```

```
            HStack {
                Text(challenge.title)
                if appState.isCompleted(challenge.id) {
                    Image(systemName: "checkmark.circle.fill")
                        .foregroundColor(.green)
                }
            }
        }
    }
}
```

🎯 User Experience Issues (First Week)

Issue: No Visual Feedback

Symptoms:

- Users tap submit, nothing happens
- No indication of app state
- Confusing user interactions

Root Cause: Missing loading states and feedback

Quick Fix:

swift

```swift
// ✅ Add loading states and feedback
struct ChallengeDetailView: View {
    let challenge: Challenge
    @State private var userCode: String = ""
    @State private var isSubmitting = false // Add loading state
    @State private var showingResult = false
    @State private var submissionResult = ""

    var body: some View {
        VStack {
            // Your existing content

            Button(action: submitSolution) {
                HStack {
                    if isSubmitting {
                        ProgressView()
                            .scaleEffect(0.8)
                    }
                    Text(isSubmitting ? "Submitting..." : "Submit Solution")
                }
            }
            .disabled(isSubmitting || userCode.trimmingCharacters(in: .whitespacesAndNewlines).isEmpty)
            .buttonStyle(.borderedProminent)
        }
        .alert("Result", isPresented: $showingResult) {
            Button("OK") { }
        } message: {
            Text(submissionResult)
        }
    }

    private func submitSolution() {
        guard !userCode.trimmingCharacters(in: .whitespacesAndNewlines).isEmpty else {
```

```
            submissionResult = "Please enter your solution before submitting."
            showingResult = true
            return
        }

        isSubmitting = true

        // Simulate processing time
        DispatchQueue.main.asyncAfter(deadline: .now() + 1.0) {
            isSubmitting = false
            submissionResult = "Solution submitted successfully!\n\nScore: 85/100\nTime: 2:30"
            showingResult = true
        }
    }
}
```

Issue: Poor Code Editing Experience

Symptoms:

- Hard to type code on mobile
- No syntax hints
- Keyboard keeps auto-correcting code

Root Cause: TextEditor not optimized for code

Quick Fix:

```
swift
```

```swift
// ✅ Better code editor experience
struct CodeEditorView: View {
    @Binding var code: String
    @FocusState private var isTextEditorFocused: Bool

    var body: some View {
        VStack(alignment: .leading, spacing: 8) {
            HStack {
                Text("Your Solution:")
                    .font(.headline)
                Spacer()
                // Line counter
                Text("\(code.components(separatedBy: .newlines).count) lines")
                    .font(.caption)
                    .foregroundColor(.secondary)
            }

            TextEditor(text: $code)
                .focused($isTextEditorFocused)
                .font(.system(.body, design: .monospaced))
                .autocapitalization(.none)
                .disableAutocorrection(true)
                .keyboardType(.asciiCapable) // Better for code
                .border(isTextEditorFocused ? Color.blue : Color.gray.opacity(0.3), width: 1)
                .frame(minHeight: 200)
                .overlay(
                    // Placeholder text when empty
                    Group {
                        if code.isEmpty {
                            VStack {
                                HStack {
                                    Text("// Enter your Swift solution here\nfunc solution() {\n    \n}")
                                        .font(.system(.body, design: .monospaced))
```

```
                        .foregroundColor(.gray.opacity(0.6))
                        .padding(8)
                    Spacer()
                }
                Spacer()
            }
            .allowsHitTesting(false)
        }
      }
    )

    // Quick insert buttons
    ScrollView(.horizontal, showsIndicators: false) {
      HStack {
        ForEach(["func ", "if ", "for ", "while ", "return ", "{}", "[]", "()"], id: \.self) { snippet in
          Button(snippet.trimmingCharacters(in: .whitespaces)) {
            insertText(snippet)
          }
          .buttonStyle(.bordered)
          .font(.caption)
        }
      }
      .padding(.horizontal)
    }
  }
}

private func insertText(_ text: String) {
  code += text
  isTextEditorFocused = true
}
}
```

🔍 Debugging Your Barebones App

Add Simple Debug Info

swift

```swift
// ✅ Add debug info to help troubleshoot
struct ContentView: View {
    @State private var challenges: [Challenge] = Challenge.sampleData

    var body: some View {
        NavigationView {
            VStack {
                // Debug info (remove in production)
                #if DEBUG
                Text("Debug: \(challenges.count) challenges loaded")
                    .font(.caption)
                    .foregroundColor(.gray)
                    .padding(.top)
                #endif

                List(challenges) { challenge in
                    NavigationLink(destination: ChallengeDetailView(challenge: challenge)) {
                        VStack(alignment: .leading, spacing: 4) {
                            Text(challenge.title)
                                .font(.headline)
                            Text(challenge.description)
                                .font(.caption)
                                .foregroundColor(.secondary)
                                .lineLimit(2)
                        }
                        .padding(.vertical, 2)
                    }
                }
            }
            .navigationTitle("CodeKata")
        }
        .onAppear {
            print("ContentView appeared with \(challenges.count) challenges") // Debug log
```

```
    }
  }
}
```

📋 MVP Health Check

After implementing fixes, verify these work:

✅ Basic Functionality Checklist

- [ ] App launches without crashes
- [ ] Challenge list displays 4 sample challenges
- [ ] Tapping challenge navigates to detail view
- [ ] Can type in text editor
- [ ] Submit button shows alert
- [ ] Navigation back to list works
- [ ] Works on both iPhone and iPad

✅ User Experience Checklist

- [ ] No obvious visual bugs
- [ ] Text is readable on all screen sizes
- [ ] Buttons are tappable (min 44pt)
- [ ] Keyboard appears/dismisses properly
- [ ] Loading states for user actions
- [ ] Error messages are helpful

🚨 Red Flags to Fix Immediately

- **App crashes on launch** → Check target membership of Swift files
- **Navigation doesn't work** → Verify NavigationView wrapper
- **Blank screens** → Check for force unwrapping optionals

- **Keyboard issues** → Add proper TextEditor configuration
- **Memory warnings** → Look for retain cycles in closures

🎯 When to Stop Adding Features

Your barebones MVP is complete when:

1. ✅ All basic functionality works
2. ✅ No crashes in normal usage
3. ✅ UI looks decent on iPhone/iPad
4. ✅ Users can complete the core flow: browse → select → code → submit

Don't add more features until this foundation is solid!

The key with MVP troubleshooting is to **fix issues immediately** as they appear. Each small problem compounds into bigger issues if left unresolved. Keep the scope minimal and make sure what you have works perfectly before adding complexity.