MU Connect – Test Plan Document

Prepared by: MU Connect team

Version History

Version	Date	Description	Author
1.0	April 10, 2025	Initial Draft	MU Connect Team
2.0	April 15, 2025	Refined the Scope & Schedule	MU Connect Team
3.0	April 18, 2025	Finalized Schedule	MU Connect Team

1. Introduction

This test plan outlines how we will verify and validate the MU Connect mobile application. MU Connect is a student-alumni networking platform tailored for Mahindra University. Our goal is to ensure the app runs smoothly, feels intuitive to users, and meets all the expected functional, security, and performance standards.

2. Objectives and Tasks

Our main objectives are:

- ➔ Confirm that each feature performs as it should
- ➔ Detect and fix bugs early
- ➔ Ensure a smooth experience across multiple devices based on the platforms we are using
- ➔ Test security, responsiveness, and usability

Key tasks include writing test cases, executing tests, fixing bugs, and documenting results.

3. Test Scope

We will test the full set of core features, including:

- ➔ User login and registration
- ➔ Profile management
- ➔ Messaging
- ➔ Job and event posting
- ➔ Forum participation
- ➔ Admin moderation tools

We will not be testing future integrations like LinkedIn or multilingual support at this stage.

4. Test Strategy

4.1 Entry and Exit Criteria

To make sure our testing process starts and ends smoothly, we've defined clear checkpoints:

Entry Criteria (When we're ready to begin testing):

- ➔ All major features of the app have been built and are available in the testing environment.
- ➔ The test environment is set up, stable, and mirrors the real usage conditions.
- ➔ Our test cases are fully written, reviewed, and good to go.
- ➔ We've gathered all the test data and tools we'll need.

Exit Criteria (When we're confident testing is complete):

- ➔ Every planned test case has been executed.
- ➔ All high-priority bugs have been either fixed or acknowledged with approved solutions.
- ➔ Our reports—test summaries, screenshots, and bug logs—are complete and accurate.
- ➔ The QA team and project leads have reviewed and signed off on everything, giving the green light.

4.2 Bug Severity and Priority Handling

To stay focused and fix the right things at the right time, we'll handle bugs based on how serious and urgent they are:

High Severity, High Priority

- ➔ Example: App crashes during login
- ➔ Action: Fix immediately before moving on

Medium Severity, High Priority

- ➔ Example: A button or feature doesn't respond
- ➔ Action: Fix soon—it affects how people use the app

Low Severity, Medium Priority

- ➔ Example: Text alignment looks off
- ➔ Action: Fix if we have time—it's not urgent, but nice to clean up

Low Severity, Low Priority

- ➔ Example: Slight color mismatch or outdated icon
- ➔ Action: Won't block anything—can fix in a later update

This way, we ensure major problems get fixed fast, while small issues are noted and handled thoughtfully.

4.3 Types of Testing

We'll apply the following types of manual testing:

- ➔ Functional Testing: Verifies that each feature (login, signup, post creation, etc.) works according to requirements.

  Example: Manually checking if a user can successfully log in with valid credentials.

- ➔ Regression Testing: Performed after changes (e.g. adding a new feature) to ensure existing functionality still works.

Example: After adding the Follow button, checking that profile viewing and posting still work.

- ➔ Integration Testing: Ensures different parts of the app (modules) work together.

  Example: Testing whether the profile updates reflect in the Connect view.

- ➔ UI/UX Testing (Visual Testing): Checks whether the user interface looks and behaves as expected.

  Example: Manually verifying that images appear in the feed and are not stretched or blurry.

- ➔ Smoke Testing: Basic tests to confirm that the app launches and major features are working.

  Example: Launch the app, navigate to all tabs, try a quick sign up and login.

- ➔ Error/Boundary Testing: Verifies how the app handles invalid inputs or unexpected user behavior.

  Example: Typing an incorrect email or password and checking if proper error is shown.

5. Test Deliverables

We'll provide:

- ➔ This test plan
- ➔ Some examples of our manual test cases
- ➔ A list of errors we encountered
- ➔ Final sign-off sheet

5.1 Manual Test Cases

We tested our features by:

- ➔ Running the app in the iOS Simulator or device
- ➔ Signing up, logging in, editing your profile
- ➔ Posting images, checking Feed and Profile
- ➔ Following users, viewing Connect page, etc

5.2 Errors

Authentication & User Management Errors

- ➔ "Missing argument for parameter 'user' in call" — Happens when trying to show the ProfileView without passing the current user object.
- ➔ "Cannot login after signing up" — Occurs due to UserDefaults not correctly saving new users or changes in the Post model.
- ➔ "currentUser not loading" — This happens when loadAllUsers() fails to decode MUUser, usually due to structure mismatches after updates.
- ➔ Couldn't restrict signup to Mahindra University email initially — Solved by implementing a regex/email format check.

Profile Page Errors

- ➔ "Loading profile..." stuck indefinitely — Caused by the users array not decoding from UserDefaults properly.

➔ Profile picture being distorted/squeezed — Fixed by using resizable, scaledToFill, and .clipShape(Circle()) attributes.
"ProfileView doesn't open" — Due to nil user or decoding failure of user data.

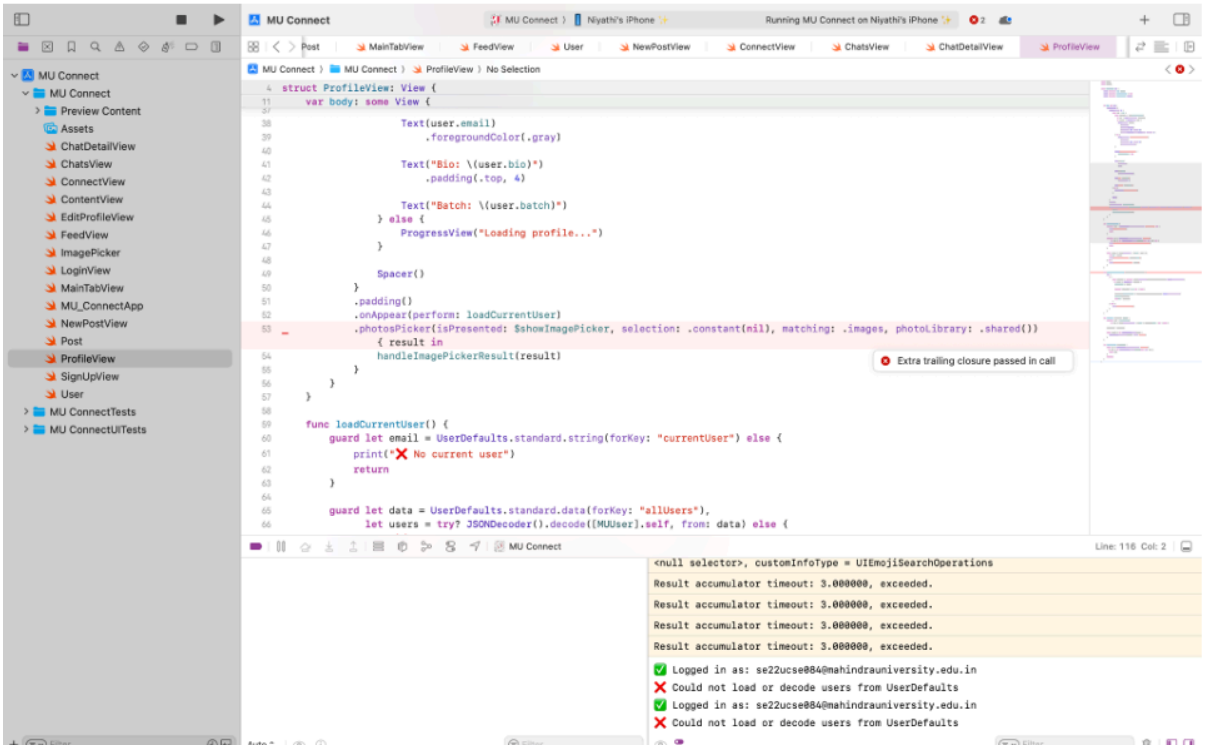
Post & Feed Errors

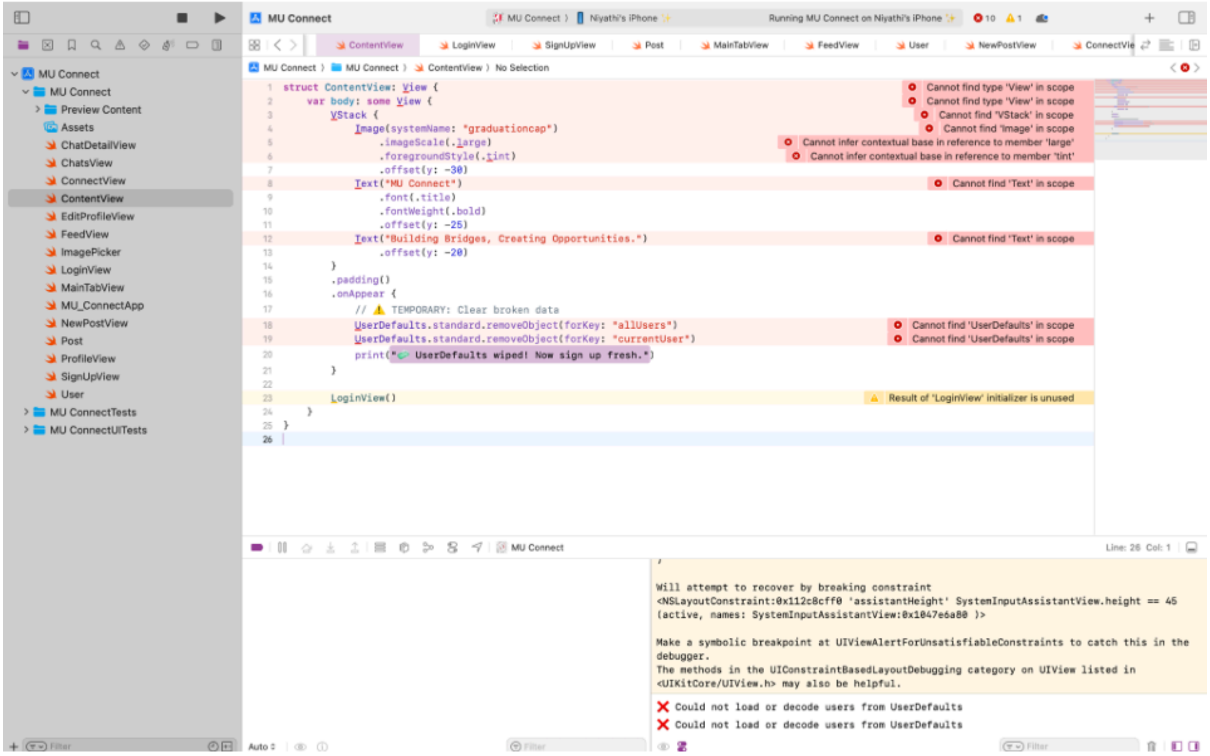
➔ "Missing argument for caption" — Happens after adding caption to Post struct but not updating the initializer call.
➔ "Feed is blank" — Can be caused by posts not saving properly or issues with Post struct being updated with Identifiable or ForEach missing id: .id.
➔ "Cannot find 'Post' or 'ContentView'" — Due to redeclaration or conflicting definitions.
➔ Trying to use Section inside ScrollView — Not correct visually or structurally in SwiftUI.
➔ Feed not showing any data — Often due to decoding errors after changes in Post model (e.g., adding caption or imageData).

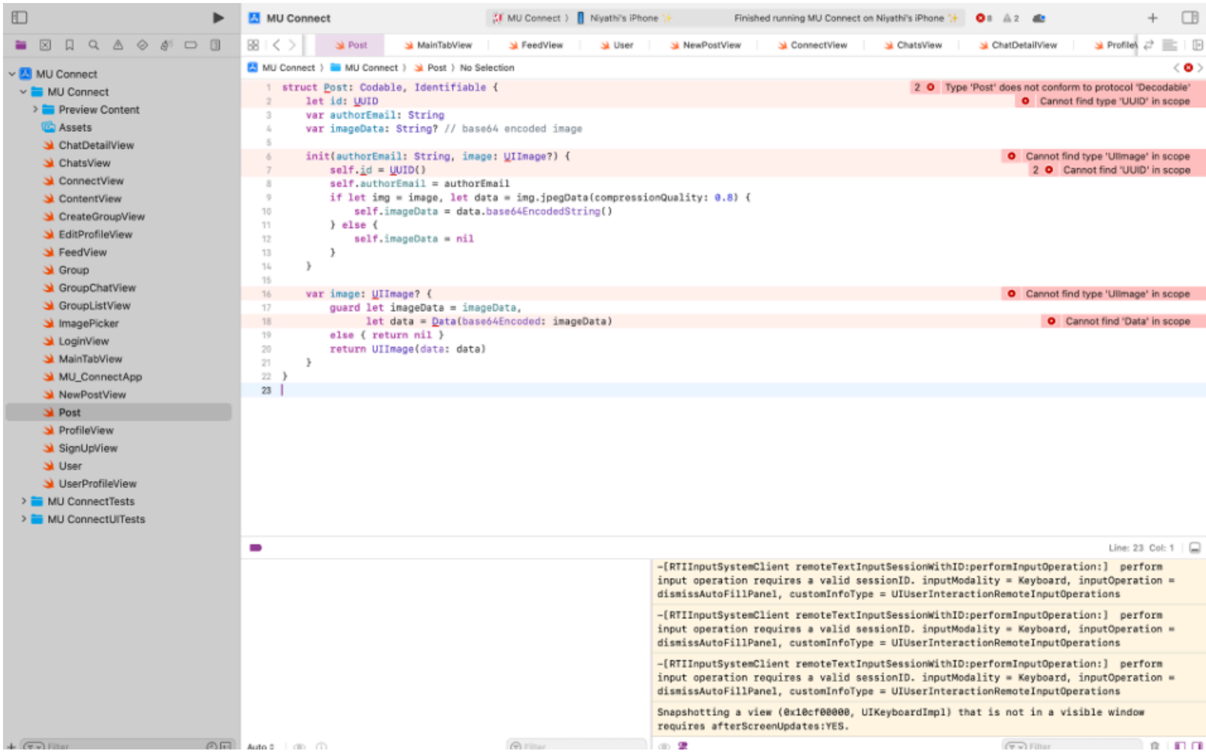
Connect Page Errors

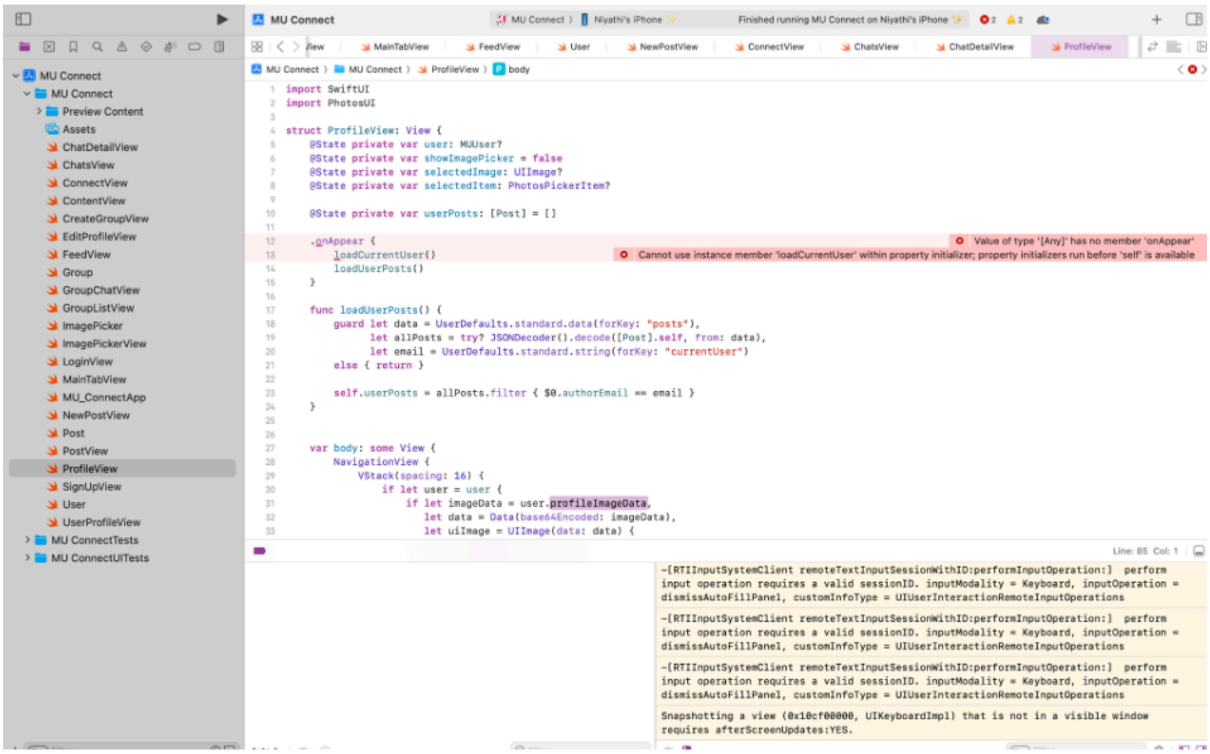
➔ No users showing — Caused by UserDefaults being cleared or decoding failures (e.g., MUUser missing expected fields).
➔ Following didn't update — Because MUUser struct didn't include following: [String] during decoding.
➔ Follow button not rendering — Current user wasn't loaded properly for comparison.
➔ General SwiftUI Issues
➔ "Invalid redeclaration of 'Post'" — Multiple files defining the same struct name.
➔ "Cannot find 'ImagePickerView' or 'ProfileView'" — File not added to your project target.
➔ "onChange(of:perform:) is deprecated" — From older iOS versions; updated to new SwiftUI lifecycle.

5.3 Screenshots of Errors we Encountered




MU Connect
Running MU Connect on Niyathi's iPhone
ContentView
struct ContentView: View {
var body: some View {
VStack {
Image(systemName: "graduationcap")
.imageScale(.large)
.foregroundStyle(.tint)
.offset(y: -30)
Text("MU Connect")
.font(.title)
.fontWeight(.bold)
.offset(y: -25)
Text("Building Bridges, Creating Opportunities.")
.offset(y: -20)
}
.padding()
.onAppear {
// ⚠️ TEMPORARY: Clear broken data
UserDefaults.standard.removeObject(forKey: "allUsers")
UserDefaults.standard.removeObject(forKey: "currentUser")
print("UserDefaults wiped! Now sign up fresh.")
}
LoginView()
}
}
Cannot find type 'View' in scope
Cannot find 'VStack' in scope
Cannot find 'Image' in scope
Cannot infer contextual base in reference to member 'large'
Cannot infer contextual base in reference to member 'tint'
Cannot find 'Text' in scope
Cannot find 'UserDefaults' in scope
Result of 'LoginView' initializer is unused
Will attempt to recover by breaking constraint
<NSLayoutConstraint:0x112c8cff0 'assistantHeight' SystemInputAssistantView.height == 45 (active, names: SystemInputAssistantView:0x1047e6a80)>
Make a symbolic breakpoint at UIViewAlertForUnsatisfiableConstraints to catch this in the debugger.
The methods in the UIConstraintBasedLayoutDebugging category on UIView listed in <UIKitCore/UIView.h> may also be helpful.
Could not load or decode users from UserDefaults
Could not load or decode users from UserDefaults


MU Connect
Finished running MU Connect on Niyathi's iPhone
Post
struct Post: Codable, Identifiable {
let id: UUID
var authorEmail: String
var imageData: String? // base64 encoded image
init(authorEmail: String, image: UIImage?) {
self.id = UUID()
self.authorEmail = authorEmail
if let img = image, let data = img.jpegData(compressionQuality: 0.8) {
self.imageData = data.base64EncodedString()
} else {
self.imageData = nil
}
}
var image: UIImage? {
guard let imageData = imageData,
let data = Data(base64Encoded: imageData)
else { return nil }
return UIImage(data: data)
}
}
Type 'Post' does not conform to protocol 'Decodable'
Cannot find type 'UUID' in scope
Cannot find type 'UIImage' in scope
Cannot find 'UUID' in scope
Cannot find 'Data' in scope
-[RTIInputSystemClient remoteTextInputSessionWithID:performInputOperation:] perform input operation requires a valid sessionID. inputModality = Keyboard, inputOperation = dismissAutoFillPanel, customInfoType = UIUserInteractionRemoteInputOperations
Snapshotting a view (0x10cf00000, UIKeyboardImpl) that is not in a visible window requires afterScreenUpdates:YES.



6. Test Environment

- ➔ Hardware: iOS devices
- ➔ Software: Xcode (IDE), Swift (language), SwiftUI (UI framework), PhotosUI (for image picking), UserDefaults (for local storage), Core Data (for structured local storage)

7. Roles and Responsibilities

Each team member has a clear role:

- ➔ Divya: Test planning and oversight
- ➔ Sadhika & Varshini: Write and run test cases
- ➔ Niyathi & Sathvick: Handle bug fixes and unit testing

Dr. Vijay Rao Duddu & Nartkannai K: Approves and reviews

8. Schedule

Here's our approximate testing timeline:

- ➔ April 18–19: Finalize all written test cases for each module (Login, Profile, Chat, Events, Jobs, Forums, Admin tools)
- ➔ April 20–21: Conduct unit testing for individual features (e.g., registration, post creation, messaging)
- ➔ April 22–23: Perform integration testing to ensure smooth coordination between modules (e.g., Login + Profile, Chat + Notifications)
- ➔ April 24–25: Execute system testing across different iOS devices
- ➔ April 26–27: Run user testing with actual MU students and alumni to get usability and experience feedback

- ➔ April 28–29: Fix reported bugs and perform regression testing to ensure nothing broke after updates
- ➔ April 30: Perform final cleanup – double-check test logs, complete any pending cases, and validate all reports
- ➔ May 1–2: Prepare and finalize test summary report, screenshots, bug logs, and approval sheets
- ➔ May 3: Ready to submit final test plan, test summary report, and app for evaluation

9. Risk Management

- ➔ Device differences might cause UI issues – we'll use responsive design and test widely
- ➔ Tight timeline – buffer days added for fixes
- ➔ Internet/API issues – mock servers will be used

10. Approval and Sign-off

Once all tests are passed and reviewed, this document will be signed off by the instructor and team leads to confirm that MU Connect is ready for release.

- Dr. Vijay Rao Duddu & Nartkannai K: Approves and reviews