CAMPUS BOOKING SYSTEM

Testing Strategy Document

Phase 1 · v1.0 · CBS-TEST-001

Purpose

This document defines the testing strategy for the Campus Booking System Phase 1 prototype and lays the groundwork for the production testing pipeline. It describes five testing levels, the tools used, test scope per epic, pass/fail criteria, and the relationship between prototype-level testing and production-level validation.

Testing Principles

Shift-left testing: requirements are testable from the moment they are written

Each user story has at least one corresponding test case defined before development begins

Prototype testing validates user flows and UI behaviour using mock data

Production testing validates system behaviour against live data and infrastructure

Accessibility testing is mandatory for every screen, not optional

Testing Levels Overview

T-01 — Unit Testing

Objective

Verify that individual Swift functions, ViewModels, and Model logic produce correct outputs for given inputs in complete isolation from the UI and data layer. Unit tests are fast, deterministic, and should run on every code commit.

Tools & Framework

XCTest framework (built into Xcode) for test execution

Swift Testing framework (@Test macro, introduced iOS 17) for modern test syntax

Mock objects and protocol-based dependency injection to isolate units under test

Scope & Test Cases

Pass Criteria

All unit tests pass with zero failures

Code coverage ≥ 80% across all ViewModel and Model files

No unit test takes longer than 0.5 seconds to execute

T-02 — Integration Testing

Objective

Verify that the ViewModel layer correctly interacts with the data layer (mock API client), and that data flows correctly from the data source through the ViewModel to the expected output state. Integration tests validate that components work correctly together, not just in isolation.

Tools & Framework

XCTest with a mock URLSession protocol conformance to simulate API responses

Custom MockDataProvider protocol injected into ViewModels for controlled data scenarios

Async/await test support using async XCTest functions

Key Integration Test Scenarios

Tutor search: mock API returns 5 tutors → ViewModel filters to available only → View receives 3 results

Room booking: ViewModel calls API to reserve room → mock API returns 200 OK → ViewModel state = .confirmed → notification scheduled

Auth token expiry: mock API returns 401 Unauthorised → AuthViewModel clears session → redirect to Login screen

Rating submission: ViewModel posts rating to mock API → response updates cached tutor aggregate score in ViewModel

Network failure: mock URLSession throws timeout error → ViewModel surfaces error state → retry option displayed

Pass Criteria

All integration tests pass against mock data layer

Each ViewModel state transition is verified (loading → success → error paths)

No direct dependencies on live network calls in the test suite

T-03 — UI / End-to-End Testing

Objective

Validate complete user flows from the perspective of a real user interacting with the iOS UI. UI tests simulate tap events, text input, scroll gestures, and navigation to verify that each user story's flow works correctly end-to-end on an iOS Simulator.

Tools & Framework

XCUITest framework — Apple's native UI testing tool for iOS

Xcode Simulator (iPhone 15 Pro, iOS 17) as the target test device

Accessibility identifiers added to all interactive elements to make them programmatically addressable

Page Object Model pattern used to organise screen interaction helpers

E2E Flow Test Cases

T-04 — Accessibility Testing

Objective

Ensure the CBS prototype is usable by students with visual, motor, or cognitive impairments. Accessibility testing is non-negotiable — it is required by both the university's digital accessibility policy and the Academic Services stakeholder (External). Every screen must pass before the prototype is considered complete.

Tools & Methods

Xcode Accessibility Inspector — identifies elements missing accessible labels or roles

iOS VoiceOver — manual screen-reading walkthrough of every screen and interactive flow

Dynamic Type scaling — test UI at smallest (xSmall) and largest (AX5) font size settings

Color Contrast Analyser — verify all text meets WCAG 2.1 AA contrast ratios (4.5:1 normal, 3:1 large)

Switch Control — verify all flows are navigable without touch input

Accessibility Test Checklist

T-05 — User Acceptance Testing (UAT)

Objective

Validate that the prototype meets real user expectations by having target users complete scripted booking flows without any guidance from the development team. UAT confirms that the system solves the problems identified during elicitation (CBS-ELIC-001).

UAT Process

Recruit 5–8 student users and 1 admin user who were not involved in development

Distribute UAT script with 5 tasks (one per core epic) — no coaching allowed

Observe users silently, note hesitations and errors without intervening

Collect System Usability Scale (SUS) survey after each session

Debrief with 5-minute structured interview on pain points and suggestions

UAT Task Script

UAT Pass Criteria

≥ 80% of users complete all 5 tasks within the time limits without assistance

Average System Usability Scale (SUS) score ≥ 68 (industry-standard ‘Good’ threshold)

No critical usability defects (user cannot complete a task at all) in any flow

All identified usability issues are documented and triaged before prototype sign-off

Document ID: CBS-TEST-001

Version: 1.0

Tech Lead: Saad Alzamzami

Phase: Phase 1

Level	Type	Tools	Scope	Owner
T-01	Unit Testing	XCTest, Swift Testing	ViewModels, Models, Utilities	Saad Alzamzami
T-02	Integration Testing	XCTest, Mock URLSession	Data layer, ViewModel + Model	Saad Alzamzami
T-03	UI / End-to-End Testing	XCUITest, Simulator	Complete user flows per epic	Bassel Taleb

Level	Type	Tools	Scope	Owner
T-04	Accessibility Testing	Xcode Accessibility Inspector, VoiceOver	All screens	Syeda Ahmed
T-05	User Acceptance Testing	Manual testing with real users	Core booking flows, UAT script	Umaya Hassan

Test ID	Component Under Test	Test Scenario	Expected Result
UT-01	AuthViewModel.login()	Valid credentials submitted	isAuthenticated = true, token stored in Keychain
UT-02	AuthViewModel.login()	Invalid password submitted	isAuthenticated = false, errorMessage populated
UT-03	BookingViewModel.filterRooms()	Capacity filter = 4, date = tomorrow	Only rooms with capacity ≥ 4 and tomorrow availability returned
UT-04	BookingViewModel.confirmBooking()	Valid room + time slot selected	Booking created, confirmationID returned, state = .confirmed
UT-05	RatingViewModel.submitRating()	Rating = 5 stars, comment = valid	Rating stored, tutor aggregate score updated
UT-06	NotificationScheduler.schedule()	Booking confirmed with future date	Two local notifications scheduled at T-24h and T-1h
UT-07	SessionToken.isExpired()	Token created 31 minutes ago	Returns true; triggers re-authentication flow
UT-08	EquipmentViewModel.checkAvailability()	Item booked for requested period	Returns .unavailable with next free slot date

Test ID	Flow	Steps	Pass Condition
UI-01	Student Registration	Launch app → Tap Register → Enter email/password → Submit	Account created screen shown
UI-02	Secure Login	Open app → Enter credentials → Tap Login	Dashboard loads with user name displayed
UI-03	Book a Tutor	Login → Tutor Booking → Search → Select tutor → Pick slot → Confirm	Confirmation screen with booking ID shown
UI-04	Book a Room	Login → Room Booking → Filter → Select room → Reserve	Room added to My Bookings
UI-05	Reserve Equipment	Login → Equipment → Browse → Reserve → Select times → Confirm	Equipment booking confirmation shown
UI-06	Submit Rating	Login → My Bookings → Past session → Rate → 5 stars → Submit	Rating submitted confirmation shown
UI-07	Admin Dashboard	Login as Admin → Navigate to Dashboard	Utilisation stats visible for all resource types

Check	Standard	Status
All images have meaningful accessibilityLabel set	WCAG 2.1 – 1.1.1 Non-text Content	Required
All buttons have descriptive accessibilityLabel (not just 'Button')	WCAG 2.1 – 1.3.1 Info and Relationships	Required
All interactive elements ≥ 44x44pt tap target	Apple HIG, WCAG 2.5.5	Required
Text contrast ratio ≥ 4.5:1 at default font size	WCAG 2.1 – 1.4.3 Contrast (AA)	Required
UI reflows correctly at AX5 Dynamic Type size	WCAG 2.1 – 1.4.4 Resize Text	Required
VoiceOver reads entire booking flow without dead ends	WCAG 2.1 – 2.1.1 Keyboard (navigation)	Required

Check	Standard	Status
Error messages are announced by VoiceOver immediately	WCAG 2.1 – 4.1.3 Status Messages	Required
No time-limited actions without extension option	WCAG 2.1 – 2.2.1 Timing Adjustable	Required

Task	Instruction Given to User	Success Criterion	Time Limit
UAT-1	Register a new account using your university email address	Account created confirmation visible	3 minutes
UAT-2	Find a tutor for Mathematics and book the first available slot	Booking confirmation with tutor name shown	3 minutes
UAT-3	Reserve study room CB-204 for tomorrow at 2pm	Room appears in My Bookings	3 minutes
UAT-4	Borrow a DSLR camera for a two-day project	Equipment reservation confirmed	3 minutes
UAT-5	Leave a star rating for your last tutor session	Rating submitted and tutor score updated	2 minutes