StudyBuddyApp — Test Plan

Test Harness (common setup for all tests)

Created fresh for each test (as in `@BeforeEach`):

Students (IDs auto-sequenced by Repository):

- 1: Alice — Courses = {CPSC 3720} — Avail = [MON 14:00–16:00]
- 2: Bob — Courses = {CPSC 3720} — Avail = [MON 15:00–17:00]
- 3: Jon — Courses = {MATH 3110} — Avail = [TUE 09:00–11:00]
- 4: Mary — Courses = {MATH 3110} — Avail = [TUE 10:00–12:00]

Sessions: none (unless created by the test)

TimeSlot

TimeSlot(DayOfWeek, start, end) – [timeSlot_constructorRejectsInvalidTimes]

Input: (MONDAY, 10:00, 10:00) and (MONDAY, 11:00, 10:00)
State (Before): N/A
Output: `IllegalArgumentException` for both
State (After): N/A

---

overlaps/intersection – [timeSlot_overlapAndIntersection]

Input:

- A = (MONDAY, 10:00–12:00)
- B = (MONDAY, 11:00–13:00)
  State (Before): N/A

Output:

- `A.overlaps(B)` = true
- `A.intersection(B)` = (MONDAY, 11:00–12:00)
  State (After): N/A

Profile / Availability

ProfileController.createProfile – [profileController_createsProfileAndNormalizesCourses]

Input: name = "Tim", courses = ["cPsC 3720", "MATH 3110"]
State (Before): Students = {1..4 as in setup}
Output: New Student (ID = 5) with:

- name = "Tim"
- courses = {"CPSC 3720", "MATH 3110"} (normalized)
  State (After):
- Students now include ID 5 with above courses.
- Other students unchanged.

AvailabilityController.addAvailability + removeAvailability – [availabilityController_addAndRemove]

Input:

- Add for Alice (ID 1): (WEDNESDAY, 10:00–11:00)
- Remove at index = previous size (i.e., the slot just added)
  State (Before):

- Alice.availability = [(MON 14:00–16:00)]
 Output:

- After add: size increments by 1

- After remove: size returns to original
 State (After):

- Alice.availability back to [(MON 14:00–16:00)]

- Others unchanged.

Repository lookups

classmatesInCourse – [classmatesInCourse_excludesSelfAndFiltersByCourse]

Input:

- Query 1: `classmatesInCourse(1, "CPSC 3720")` (Alice)

- Query 2: `classmatesInCourse(1, "MATH 3110")` (Alice)
 State (Before): Students as in setup
 Output:

- Query 1: List = [Bob] (size = 1)

- Query 2: List = [Jon, Mary] (size = 2) — method returns classmates in the course regardless of caller's enrollment
 State (After): No state change

Sessions

Create → Join → Confirm – [session_createJoinConfirm_flow]

Input:

1. Create session `S1`:
    - course = "CPSC 3720"
    - time = (MONDAY, 15:00–16:00)
    - participants = [Alice]
2. Join `S1` as Bob
3. Confirm `S1` as Alice, then Bob
   State (Before): Sessions = {}
   Output:

- After create: `S1` exists; participants = {Alice}; confirmed = {}
- After join: participants = {Alice, Bob}; confirmed = {}
- After confirmations: confirmed = {Alice, Bob}; `isFullyConfirmed()` = true
  State (After):
- Sessions = { S1: course=CPSC 3720, time=MON 15:00–16:00, participants={1,2}, confirmed={1,2} }

Search by Course / by Student Name – [session_searchByCourseAndByName]

Input: Create two sessions:

- `S1`: CPSC 3720, (MON 15:00–16:00), participants={Alice, Bob}
- `S2`: MATH 3110, (TUE 10:30–11:30), participants={Jon, Mary}
  Then:
- `searchByCourse("cpsc 3720")`
- `searchByStudentName("ar")` (matches “Mary”)
  State (Before): Sessions = {}
  Output:

- By course: returns [S1]
- By name: returns [S2]
  State (After):
- Sessions persist: {S1, S2}

Suggestions

suggestMatches – [suggestMatches_findsOverlapWindows]

Input: `suggestMatches(1, "CPSC 3720")` (Alice)
State (Before):

- Alice.availability = (MON 14:00–16:00)
- Bob.availability = (MON 15:00–17:00)
  Output:
- Suggestions map includes key = Bob, value contains overlap (MON 15:00–16:00)
  State (After): No state change

Notes / Assumptions

- Student IDs are assigned incrementally starting at 1 per new Repository instance.
- Course names are normalized to uppercase for storage/lookup.
- `classmatesInCourse` returns peers in a course regardless of whether the caller is enrolled in that course (as used by the test).
- Session confirmation is per-participant; a session is “fully confirmed” only when all participants have confirmed.