Cal Poly Pomona

Advanced Software Engineering (CS 5800.01)

# Project Part 3 (Unit and Integration Tests)


## Project Partners

1. Talha Ahmed
2. Noah Loke
3. Eric Dechant Trinh


Submitted to: Professor Nima Davarpanah

Dated: Monday, October 27, 2025

## Table of Contents

# Part 1 – Unit Test

## User (Sign-up)

### Method Signature

void signUp(String name, String email, String password)

### Performance / Non-Functional Notes

NFR-1. The user profile creation process shall complete within one second for 95% of the attempts under normal server load.

NFR-3. The system shall store the passwords using the hashing algorithm SHA-512.

### Fields

1. id: UUID
2. name: String
3. email: String

### Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.01 | UC-1 | SignUp_Success_Basic | Registers a new user with valid name, well-formed email, and strong password. | Positive | Pass | None | None | Email not previously registered. Password policy enforced by PasswordPolicy service. | name: non-empty; email: RFC-5322 simple; password: >=8 chars incl. upper/lower/digit/special characters | None | name length=1; name length=255; password length=8; password length=64 |
| 1.02 | UC-1 | SignUp_Fail_DuplicateEmail | Attempts to register with an email that already exists for another user. | Negative/Exception | Fail | DuplicateEmailException | Email already registered. | UserRepository contains an existing user with same normalized email. | name valid; email normalized case-insensitively | email collides post-trim/lowercase normalization | emails 'Test@Ex.com' vs ' test@ex.COM ' |
| 1.03 | UC-1 | SignUp_Fail_InvalidEmailFormat | Email lacks '@' or domain parts. | Negative/Exception | Fail | InvalidEmailFormatException | Email format is invalid. | None | None | email missing '@', multiple '@', invalid TLD, leading/trailing spaces only | email '', 'foo', 'foo@', '@bar.com', 'foo@bar' |
| 1.04 | UC-1 | SignUp_Fail_WeakPassword | Password does not meet complexity rules. | Negative/Exception | Fail | WeakPasswordException | Password does not meet complexity requirements. | PasswordPolicy defines min length 8, requires upper, lower, digit, special characters. | None | password too short, missing digit, missing uppercase, missing lowercase, missing special character | password 'abc', 'abcdefgh', 'Abcdefgh', 'abcdefg1' |
| 1.05 | UC-1 | SignUp_Fail_BlankName | Name is null, empty, or whitespace. | Negative/Exception | Fail | InvalidNameException | Name cannot be blank. | None | None | name null, '', ' ' | name length=0 |
| 1.06 | UC-1 | SignUp_Success_NameBounds | Accepts minimum and maximum allowed name lengths. | Boundary | Pass | None | None | Max name length constant = 255. | name length=1 and 255 | None | 1, 255, 256 (should fail) |
| 1.07 | UC-1 | SignUp_Success_TrimNormalize | Trims leading/trailing whitespace and normalizes email to lowercase. | Positive | Pass | None | None | Normalization strategy: email lowercased; name trimmed only; password trimmed only. | email ' Foo@Bar.com ' -> 'foo@bar.com'; name ' Alice ' -> 'Alice' | None | spaces around inputs |
| 1.08 | UC-1 | SignUp_Fail_NullArgs | Any argument is null. | Negative/Exception | Fail | NullArgumentException | Argument cannot be null: {param}. | None | None | name/email/password = null | null |
| 1.09 | UC-1 | SignUp_Success_MaxPasswordLength | Allows very long but valid passwords up to 64 chars. | Boundary | Pass | None | | Max password length = 64. | password length=64 valid | password length=65 -> fail | 64, 65 |
| 1.1 | UC-1 | User_SignUp_WhenRepositoryFails | Attempts to register user when repository ran into failure. | Negative/Exception | Fail | PersistenceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# User (Login)

## Method Signature

boolean login(String email, String password)

## Performance / Non-Functional Notes

NFR-2. The user login process shall complete within three seconds for 90% of the attempts under normal server load.

## Fields

1. email: String
2. password: String

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.11 | UC-2 | Login_Success | Logs in with correct credentials. | Positive | Pass | None | None | User exists; stored password hash matches. | email valid; password correct | None | exact match vs case-insensitive email |
| 1.12 | UC-2 | Login_Fail_WrongPassword | Rejects wrong password. | Negative/Exception | Fail | AuthenticationException | Invalid email or password. | User exists; wrong password provided. | None | password incorrect | min vs off-by-one character |
| 1.13 | UC-2 | Login_Fail_NoSuchUser | Email not registered. | Negative/Exception | Fail | AuthenticationException | Invalid email or password. | Repository returns null for user lookup. | None | unknown email | None |
| 1.14 | UC-2 | Login_Fail_InvalidEmailFormat | Invalid email format is rejected before lookup. | Negative/Exception | Fail | InvalidEmailFormatException | Email format is invalid. | None | None | malformed email | edge cases as in signup |
| 1.15 | UC-2 | Login_Fail_NullOrBlank | Null or blank email/password. | Negative/Exception | Fail | NullArgumentException | Argument cannot be null or blank: {param}. | None | None | null, '', ' ' | None |
| 1.16 | UC-2 | Login_Success_EmailNormalization | Trims and lowercases email before lookup. | Positive | Pass | None | None | Same normalization strategy as signUp. | email ' Foo@Bar.COM ' works | None | spaces and case |
| 1.17 | UC-2 | Login_Success_PasswordNormalization | Trims password before lookup. | Positive | Pass | None | None | Same normalization strategy as signUp. | password ' skhJ9)*' | None | spaces |
| 1.18 | UC-2 | Login_WhenRepositoryFails | Attempts to login user when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# User (Add Course)

## Method Signature

Course addCourse(String code, String name)

## Performance / Non-Functional Notes

NFR-4. The system shall validate the uniqueness of 10000+ codes of courses within a second under normal server load for 95% of the attempts.

## Fields

1. code: String

2. name: String

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.19 | UC-3 | AddCourse_Success | Adds a new course with a unique code for the user. | Positive | Pass | None | None | User has zero or more courses; code must be unique per user. | code alphanumeric (e.g., CS101); name non-empty | None | code length=1; name length=1 |
| 1.2 | UC-3 | AddCourse_Fail_DuplicateCode_SameUser | Rejects adding a course with a code that already exists for this user. | Negative/Exception | Fail | DuplicateCourseCodeException | Course code must be unique per user. | User already owns course with code 'CS101'. | None | code collides ignoring case and whitespace | CS101 vs ' cs101 ' |
| 1.21 | UC-3 | AddCourse_Success_DuplicateCode_DifferentUser | Allows same course code for a different user (per-user uniqueness). | Positive | Pass | None | None | Another user has 'CS101' but current user does not. | code 'CS101'; name 'Intro to CS' | None | None |
| 1.22 | UC-3 | AddCourse_Fail_BlankCode | Code is null, empty, or whitespace. | Negative/Exception | Fail | InvalidCourseCodeException | Course code cannot be blank. | None | None | code null, '', ' ' | length=0 |
| 1.23 | UC-3 | AddCourse_Fail_BlankName | Course name is null or blank. | Negative/Exception | Fail | InvalidCourseNameException | Course name cannot be blank. | None | None | name null, '', ' ' | length=0 |
| 1.24 | UC-3 | AddCourse_Success_CodeNormalization | Trims and uppercases course code for comparison and storage. | Positive | Pass | None | None | Code canonical form is uppercased; spaces trimmed. | input ' cs101 ' stored as 'CS101' | None | leading/trailing spaces |
| 1.25 | UC-3 | AddCourse_Boundaries | Accepts min/max lengths for code and name; rejects beyond max. | Boundary | Pass/Fail | InvalidLengthException | Field exceeds maximum length. | Max code=16, max name=255. | code len 1 & 16; name len 1 & 255 | code len 17; name len 256 | 1,16,17 ; 1,255,256 |
| 1.26 | UC-3 | AddCourse_WithInvalidCodePattern | Rejects adding a course with a code that voilates the code pattern. | Negative/Exception | Fail | InvalidCourseCodeException | Course code must follow the code pattern. | code must be unique per user; code must not follow pattern 'CS101'. | code has the pattern 'CS101', 'cs101'. | None | None |
| 1.27 | UC-3 | AddCourse_WhenRepositoryFails | Attempts to add course when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# User (Add To-Do List)

## Method Signature

ToDoList createToDoList(String name)

## Performance / Non-Functional Notes

None

## Fields

1. name: String

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.28 | UC-7 | CreateToDoList_Success | Creates a to-do list with a non-empty name. | Positive | Pass | None | None | User can own multiple lists; no uniqueness constraint assumed. | name non-empty | None | name length=1 |
| 1.29 | UC-7 | CreateToDoList_Fail_BlankName | Null or blank name rejected. | Negative/Exception | Fail | InvalidToDoListNameException | To-do list name cannot be blank. | None | None | null, '', ' ' | name length=0 |
| 1.3 | UC-7 | CreateToDoList_Success_TrimmedName | Trims name before storage. | Positive | Pass | None | None | Trimming only; case preserved. | input ' Errands ' -> 'Errands' | None | spaces |
| 1.31 | UC-7 | CreateToDoList_WhenRepositoryFails | Attempts to create to-do list when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# UserCredential (Set Password)

## Method Signature

void setPassword(String passwordHash)

## Performance / Non-Functional Notes

NFR-3. The system shall store the passwords using the hashing algorithm SHA-512.

## Fields

1. passwordHash: String

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2.01 | UC-1 | SetPassword_Success_ExactSHA512Hex | Stores a 128-char lowercase hex SHA-512 password hash successfully. | Positive | Pass | None | None | algorithm == 'SHA-512'; hash must be 128 hex chars | 128-char lowercase hex | None | len=128 |
| 2.02 | UC-1 | SetPassword_Fail_Null | Null passwordHash throws NullArgumentException. | Negative/Exception | Fail | NullArgumentException | passwordHash cannot be null. | None | None | null | None |
| 2.03 | UC-1 | SetPassword_Fail_BlankOrWhitespace | Blank or whitespace-only passwordHash is rejected. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash cannot be blank. | None | None | '', ' ' | len=0 |
| 2.04 | UC-1 | SetPassword_Fail_InvalidHexChars | Rejects passwordHash containing non-hex characters. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash must be hexadecimal. | Must match ^[0-9a-fA-F]+$ | None | contains non-hex char | None |
| 2.05 | UC-1 | SetPassword_Fail_WrongLength_Short | Rejects hash shorter than 128 hex chars. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash length must be 128. | None | None | len<128 | len = 127 |
| 2.06 | UC-1 | SetPassword_Fail_WrongLength_Long | Rejects hash longer than 128 hex chars. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash length must be 128. | None | None | len>128 | len = 129 |
| 2.07 | UC-1 | SetPassword_Success_UppercaseHexAccepted | Accepts uppercase hex by normalizing to lowercase before storage. | Positive | Pass | None | None | Canonical storage: lowercase hex only. | 128-char uppercase hex | None | case mix |
| 2.08 | UC-1 | SetPassword_Success_IdempotentSetSameHash | Setting the same hash twice is idempotent. | Positive | Pass | None | None | algorithm immutable | same value twice | None | repeat |
| 2.09 | UC-1 | SetPassword_Fail_LogsNoSecrets | Ensures no secret values logged. | Security/NF | Pass | None | None | Logging redacts inputs | None | invalid hex/length | None |
| 2.1 | UC-1 | SetPassword_Success_AlgorithmImmutable | algorithm field remains 'SHA-512'. | Positive | Pass | None | None | algorithm constant | valid hash | None | None |
| 2.11 | UC-1 | SetPassword_WhenRepositoryFails | Attempts to set password when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# UserCredential (Verify Password)

## Method Signature

boolean verify(String passwordHash)

## Performance / Non-Functional Notes

NFR-2. The user login process shall complete within three seconds for 90% of the attempts

under normal server load.

## Fields

1. passwordHash: String

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2.12 | UC-2 | Verify_Success_MatchingHash | Returns true when provided hash matches stored hash. | Positive | Pass | None | None | Stored hash normalized lowercase | same hash | None | None |
| 2.13 | UC-2 | Verify_False_NonMatchingHash | Returns false for non-matching hash. | Positive-Negative | Pass | None | None | Stored hash set | valid hex, different value | None | Hamming distance 1 |
| 2.14 | UC-2 | Verify_Fail_Null | Null hash input throws NullArgumentException. | Negative/Exception | Fail | NullArgumentException | passwordHash cannot be null. | Stored hash set | None | null | None |
| 2.15 | UC-2 | Verify_Fail_BlankOrWhitespace | Blank or whitespace-only hash is rejected. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash cannot be blank. | Stored hash set | None | '', ' ' | len=0 |
| 2.16 | UC-2 | Verify_Fail_InvalidHexChars | Rejects non-hex characters. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash must be hexadecimal. | Stored hash set | None | non-hex | None |
| 2.17 | UC-2 | Verify_Fail_WrongLength_Short | Rejects length shorter than 128. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash length must be 128. | Stored hash set | None | len<128 | len = 127 |
| 2.18 | UC-2 | Verify_Fail_WrongLength_Long | Rejects length longer than 128. | Negative/Exception | Fail | InvalidPasswordHashException | passwordHash length must be 128. | Stored hash set | None | len>128 | len = 129 |
| 2.19 | UC-2 | Verify_Success_UppercaseInput | Uppercase input matches lowercase stored. | Positive | Pass | None | None | Stored lowercase hash | uppercase input | None | case mix |
| 2.2 | UC-2 | Verify_Security_ConstantTime_Compares | Compare time independent of first difference. | Security/Perf | Pass | None | None | Constant-time algorithm | two 128-char hex | None | distance 1 vs 64 |
| 2.21 | UC-2 | Verify_Behavior_WithoutPriorSet | verify() before setPassword(). | Behavior | Fail | IllegalStateException | Password hash not initialized. | passwordHash unset | None | None | None |
| 2.22 | UC-2 | Verify_Immutability_AlgorithmUnaffected | verify must not mutate algorithm. | Positive | Pass | None | None | algorithm invariant 'SHA-512' | valid hash | None | None |
| 2.23 | UC-2 | Verify_Performance_ManyChecks | 10k verify calls performance. | Performance | Pass | None | None | JVM warmup done | 128-char hex | None | n=10000 |
| 2.24 | UC-2 | VerifyPassword_WhenRepositoryFails | Attempts to verify password when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# Course (Get Progress)

## Method Signature

int progress()

## Performance / Non-Functional Requirement

None

## Fields

None

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3.01 | UC-3 | Progress_NoTasks_ReturnsZero | Progress is 0 when there are no tasks. | Positive | Pass | None | None | Course tasks list initially empty. | tasks count=0 | None | None |
| 3.02 | UC-3 | Progress_Partial | Returns percentage of completed tasks, rounded to nearest integer. | Positive | Pass | None | None | Progress = round(completed/total*100). | 3/5 completed => 60% | None | rounding at .5 up |
| 3.03 | UC-3 | Progress_AllCompleted | Returns 100 when all tasks are completed. | Positive | Pass | None | None | All tasks have completed=true. | None | None | None |
| 3.04 | UC-3 | Progress_RoundingBoundaries | Validates rounding near thresholds (e.g., 2/3=67%). | Boundary | Pass | None | None | Rounding rule: nearest integer, .5 rounds up. | 2/3 => 67, 1/3 => 33 | None | fractions around .5 |
| 3.05 | UC-3 | Progress_Performance_10kTasks | Computes progress efficiently with 10,000 tasks. | Performance | Pass | None | None | Acceptable time <200ms in test env. | None | None | n=10,000 |
| 3.06 | UC-3 | Progress_WhenRepositoryFails | Attempts to query course's tasks when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# Course (Add Course's Task)

## Method Signature

CourseTask addTask(String name, Date deadline, String description)

## Performance / Non-Functional Requirement

None

## Fields

1. name: String
2. deadline: Date
3. description: String

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3.07 | UC-4 | AddCourseTask_Success | Adds a task with valid name, future deadline, and optional description. | Positive | Pass | None | None | Deadline must be >= now + 1min. | name non-empty; deadline future; desc optional | None | deadline = now+1min (edge) |
| 3.08 | UC-4 | AddCourseTask_Fail_PastDeadline | Rejects tasks whose deadline is in the past. | Negative/Exception | Fail | InvalidDeadlineException | Deadline cannot be in the past. | System clock stubbed to fixed instant. | None | deadline < now | now - 1ms; now - 1 day |
| 3.09 | UC-4 | AddCourseTask_Fail_BlankName | Null or blank task name is rejected. | Negative/Exception | Fail | InvalidTaskNameException | Task name cannot be blank. | None | None | null, '', ' ' | None |
| 3.1 | UC-4 | AddCourseTask_Success_LongDescription | Allows long descriptions up to 2000 chars; rejects longer. | Boundary | Pass/Fail | InvalidLengthException | Description exceeds maximum length. | Max desc len=2000. | desc len=0, 2000 | desc len=2001 | 0, 2000, 2001 |
| 3.11 | UC-4 | AddCourseTask_Info_DuplicateNameAllowed | Documents behavior: duplicate task names are allowed within a course. | Informational | Pass | None | None | No uniqueness constraint specified in UML. | two tasks named 'HW1' | None | None |
| 3.12 | UC-4 | AddCourseTask_WhenRepositoryFails | Attempts to add course's task when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# CourseTask (Update Course's Task Progress)

## Method Signature

void updateProgress(int value)

## Performance / Non-Functional Requirement

NFR-5. The system shall update a course's task completion percentage within 500 milliseconds for 95% of the attempts.

## Fields

1. value: int

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4.01 | UC-5 | UpdateProgress_Success_ToZero | Sets progress to 0 and status=TODO. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | value=0 | None | 0,100 |
| 4.02 | UC-5 | UpdateProgress_Success_ToMiddle | Sets progress mid (e.g., 55) -> IN_PROGRESS. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | value in (1..99) | None | 1,99 |
| 4.03 | UC-5 | UpdateProgress_Success_ToHundred | Sets progress to 100 -> COMPLETED. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | value=100 | None | value = 100 |
| 4.04 | UC-5 | UpdateProgress_Fail_Negative | Rejects values below 0. | Negative/Exception | Fail | InvalidProgressException | progress must be between 0 and 100. | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | None | value<0 | value = -1 |
| 4.05 | UC-5 | UpdateProgress_Fail_OverHundred | Rejects values above 100. | Negative/Exception | Fail | InvalidProgressException | progress must be between 0 and 100. | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | None | value>100 | value = 101 |
| 4.06 | UC-5 | UpdateProgress_Boundary_ZeroThenOne | Boundary bump 0+1 sets IN_PROGRESS. | Boundary | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | value=1 (after 0) | None | 0+1 |
| 4.07 | UC-5 | UpdateProgress_Boundary_99Then100 | Boundary bump 99+100 sets COMPLETED. | Boundary | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | value=100 (after 99) | None | 99+100 |
| 4.08 | UC-5 | UpdateProgress_Success_Idempotent_SetSameValue | Setting same value is idempotent. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | value == currentProgress | None | e.g., current=55, value=55 |
| 4.09 | UC-5 | UpdateProgress_Success_AfterCompleted_To100 | After completion, setting 100 again is idempotent. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | status=COMPLETED and value=100 | None | 100+100 |
| 4.1 | UC-5 | UpdateProgress_Performance_ThousandUpdates | 1000 sequential valid increases show no leaks or nonlinear slowdown. | Performance | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | sequence 0→…→100 | None | n=1000 updates |
| 4.11 | UC-5 | UpdateProgress_WhenRepositoryFails | Attempts to update course's task progress when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# CourseTask (Mark Course's Task Complete)

## Method Signature

void markComplete()

## Performance / Non-Functional Requirement

NFR-6. The system shall mark a to-do list task's completion within 500 milliseconds for 95% of the attempts.

## Fields

None

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4.12 | UC-6 | MarkComplete_Success_FromTodo | From TODO, markComplete sets progress=100 & COMPLETED. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | initial progress=0, status=TODO | None | 0→100 |
| 4.13 | UC-6 | MarkComplete_Success_FromInProgress | From IN_PROGRESS, markComplete sets progress=100 & COMPLETED. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | initial progress in (1..99), status=IN_PROGRESS | None | 55→100 |
| 4.14 | UC-6 | MarkComplete_Success_Idempotent_WhenAlreadyCompleted | If already COMPLETED, calling markComplete is idempotent. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | initial progress=100, status=COMPLETED | None | 100→100 |
| 4.15 | UC-6 | MarkComplete_DoesNotAlterDeadline | markComplete must not mutate deadline. | Positive | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | deadline unchanged | None | None |
| 4.16 | UC-6 | MarkComplete_Performance_O1 | Executes in O(1) with no heavy work. | Performance | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | None | None | None |
| 4.17 | UC-6 | NoSensitiveLogging_OnFailures | No sensitive fields (name/description) or internal state are logged on failures. | Security/NF | Pass | None | None | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; After COMPLETED, updateProgress(v<100) illegal; updateProgress non-decreasing. | None | Any invalid updateProgress input | None |
| 4.18 | UC-6 | MarkComplete_WhenRepositoryFails | Attempts to mark course's task complete when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# ToDoList (Add Task)

## Method Signature

void addTask(String description, Date deadline)

## Performance / Non-Functional Requirement

None

## Fields

1. description: String
2. deadline: Date

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5.01 | UC-8 | AddTask_Success_Minimal | Short non-empty description; deadline slightly in future. | Positive | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | desc='Buy milk'; deadline=now+1m | None | desc len=1; now+1m |
| 5.02 | UC-8 | AddTask_Success_Trimmed | Trims leading/trailing whitespace; preserves internal spaces. | Positive | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | desc=' Call mom ' -> 'Call mom' | None | leading/trailing spaces |
| 5.03 | UC-8 | AddTask_Success_MaxLength | Accepts description at 500-char limit. | Boundary | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | desc len=500; deadline=tomorrow | None | len=500 |
| 5.04 | UC-8 | AddTask_Success_Unicode | Supports Unicode text (emojis/diacritics). | Positive | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | desc='Finish résumé ✨' | None | unicode |
| 5.05 | UC-8 | AddTask_Success_DeadlineEqualsNow | Allows deadline equal to 'now' (>= now). | Boundary | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | deadline==now | None | now |
| 5.06 | UC-8 | AddTask_Fail_NullDescription | Null description rejected. | Negative/Exception | Fail | NullArgumentException | description cannot be null. | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | None | desc=null | N/A |
| 5.07 | UC-8 | AddTask_Fail_BlankDescription | Blank or whitespace-only description rejected. | Negative/Exception | Fail | InvalidDescriptionException | description cannot be blank. | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | None | desc='', ' ' | len=0 |
| 5.08 | UC-8 | AddTask_Fail_OverMaxLength | Description > 500 chars rejected. | Negative/Exception | Fail | InvalidLengthException | description exceeds maximum length (500). | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | None | desc len>500 | len=501 |
| 5.09 | UC-8 | AddTask_Fail_NullDeadline | Null deadline rejected. | Negative/Exception | Fail | NullArgumentException | deadline cannot be null. | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | None | deadline=null | N/A |
| 5.1 | UC-8 | AddTask_Fail_PastDeadline_Tiny | Deadline 1ms in the past rejected. | Negative/Exception | Fail | InvalidDeadlineException | deadline cannot be in the past. | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | None | deadline<now | now-1ms |
| 5.11 | UC-8 | AddTask_Fail_PastDeadline_Big | Deadline days in the past rejected. | Negative/Exception | Fail | InvalidDeadlineException | deadline cannot be in the past. | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | None | deadline<now | now-2d |
| 5.12 | UC-8 | AddTask_Behavior_DuplicateDescriptionsAllowed | Duplicate descriptions allowed within the same list. | Behavior | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | two identical descriptions | None | N/A |
| 5.13 | UC-8 | AddTask_ReturnsTask_PostNormalization | Returned task contains trimmed description and exact deadline. | Positive | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | output reflects normalization | None | N/A |
| 5.14 | UC-8 | AddTask_Timezone_DSTBoundary | Validates against DST transition to avoid off-by-one-hour errors. | Boundary | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | deadline in DST transition hour | None | DST change window |
| 5.15 | UC-8 | AddTask_Performance_Batch_1000 | Add 1000 tasks without O(n^2) behavior. | Performance | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | n=1000 valid tasks | None | n=1000 |
| 5.16 | UC-8 | AddTask_Security_NoPIIInLogs | Failures never log raw description or PII. | Security/NF | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | logger redaction | invalid inputs | N/A |
| 5.17 | UC-8 | AddTask_I18N_NormalizeNFC | Normalize Unicode to NFC to avoid visually identical but unequal strings. | Behavior | Pass | None | None | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | compose 're\u0301sume\u0301' to 'résumé' | None | unicode compose/decompose |
| 5.18 | UC-8 | AddTask_WhenRepositoryFails | Attempts to add to-do list task when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# ToDoListTask (Mark Complete)

## Method Signature

void markComplete()

## Performance / Non-Functional Requirement

NFR-6. The system shall mark a to-do list task's completion within 500 milliseconds for 95% of the attempts.

## Fields

None

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6.01 | UC-9 | MarkComplete_Success_FromTODO | When status is TODO, markComplete sets status to COMPLETED. | Positive | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | initial status=TODO | None | TODO→COMPLETED |
| 6.02 | UC-9 | MarkComplete_Success_FromInProgress | When status is IN_PROGRESS, markComplete sets status to COMPLETED. | Positive | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | initial status=IN_PROGRESS | None | IN_PROGRESS→COMPLETED |
| 6.03 | UC-9 | MarkComplete_Success_Idempotent_WhenAlreadyCompleted | If already COMPLETED, markComplete is a no-op (remains COMPLETED). | Positive | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | initial status=COMPLETED | None | COMPLETED→COMPLETED |
| 6.04 | UC-9 | MarkComplete_DoesNotAlterDescription | markComplete must not mutate the description. | Positive | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | any non-empty description | None | None |
| 6.05 | UC-9 | MarkComplete_DoesNotAlterDeadline | markComplete must not mutate the deadline. | Positive | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | deadline any instant | None | None |
| 6.06 | UC-9 | MarkComplete_Allowed_WhenDeadlineInFuture | Completing ahead of the deadline is allowed. | Behavior | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | deadline > now | None | now vs now+1h |
| 6.07 | UC-9 | MarkComplete_Allowed_WhenDeadlineInPast | Completing after the deadline is allowed (no exception). | Behavior | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | deadline < now | None | now-1d |
| 6.08 | UC-9 | MarkComplete_Timezone_DSTBoundary_NoChange | Completing around DST transitions does not shift stored deadline. | Boundary | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | deadline during DST change window | None | DST transition window |
| 6.09 | UC-9 | MarkComplete_Security_NoPIIInLogs | No PII (description/deadline) is logged when markComplete is invoked. | Security/NF | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | logger redaction in place | None | None |
| 6.1 | UC-9 | MarkComplete_Reentrancy_NoDuplicateSideEffects | Multiple rapid calls to markComplete leave a consistent COMPLETE state (no race side effects). | Performance/Robustness | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | two concurrent/rapid calls | None | None |
| 6.11 | UC-9 | MarkComplete_NoExternalIO | markComplete performs no external I/O or repository writes by itself (domain object only). | Behavior | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | pure domain method | None | None |
| 6.12 | UC-9 | MarkComplete_Robust_DespiteNullOptionalFields | If description were null due to upstream bug, markComplete still completes without NPE (optional guard). | Negative/Robustness | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | description may be null (defensive) | null is tolerated | None |
| 6.13 | UC-9 | MarkComplete_Performance_O1 | Executes in O(1) and completes under 1ms on CI baseline. | Performance | Pass | None | None | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; no logging of PII. | None | None | None |
| 6.14 | UC-9 | MarkComplete_WhenRepositoryFails | Attempts to mark to-do list task complete when repository ran into failure. | Negative/Exception | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# Calendar (Show Items for Month)

## Method Signature

List<CalendarItem> itemsForMonth(User user)

## Performance / Non-Functional Requirement

NFR-7. The system shall load the monthly calendar view within three seconds for 200+ deadlines for 95% of the attempts.

## Fields

1. user: User

# Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 7.01 | UC-10 | ItemsForMonth_Success_NotItems | Returns empty list when user has no items in the month. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | valid user, month has 0 items | None | N/A |
| 7.02 | UC-10 | ItemsForMonth_Success_SingleDayEventsOnly | Returns single-day events wholly inside the month. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | events with start==end in target month | None | mid-month day |
| 7.03 | UC-10 | ItemsForMonth_Success_MultiDayWithinMonth | Returns multi-day events entirely within the target month. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | start>=firstOfMonth && end<=lastOfMonth | None | N/A |
| 7.04 | UC-10 | ItemsForMonth_Success_SpanningItemFromPrevMonth | Includes events that start in previous month but end in target month. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | start< firstOfMonth && end>=firstOfMonth | None | cross-month start |
| 7.05 | UC-10 | ItemsForMonth_Success_SpanningItemToNextMonth | Includes events that start in target month and end in next month. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | start<=lastOfMonth && end>lastOfMonth | None | cross-month end |
| 7.06 | UC-10 | ItemsForMonth_Success_FiltersByUserOwnership | Returns only items owned by or shared with the user, excludes others. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | mixed ownership dataset | None | N/A |
| 7.07 | UC-10 | ItemsForMonth_Success_SortedAscending | Results are sorted by start time ascending; stable ordering on ties. | Positive | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | multiple items with varying starts | None | two on same start |
| 7.08 | UC-10 | ItemsForMonth_Boundary_FirstAndLastDayInclusive | Events starting on 1st or ending on last day are included. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | start==firstOfMonth or end==lastOfMonth | None | 1st/last of month |
| 7.09 | UC-10 | ItemsForMonth_Boundary_FebNonLeap | Handles February 28 for non-leap year correctly. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | year not leap, month=2 | None | Feb 28 last day |
| 7.1 | UC-10 | ItemsForMonth_Boundary_FebLeapYear | Handles February 29 for leap year correctly. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | year leap, month=2 | None | Feb 29 included |
| 7.11 | UC-10 | ItemsForMonth_Boundary_Month With30Days | Handles months with 30 days accurately. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | month in {4,6,9,11} | None | 30th last day |
| 7.12 | UC-10 | ItemsForMonth_Boundary_Month With31Days | Handles months with 31 days accurately. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | month in {1,3,5,7,8,10,12} | None | 31st last day |
| 7.13 | UC-10 | ItemsForMonth_Fail_NullUser | Null user is rejected. | NegativeException | Fail | NullArgumentException | user cannot be null. | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | None | user=null | N/A |
| 7.14 | UC-10 | ItemsForMonth_Fail_InvalidMonthLow | Month < 1 is rejected. | NegativeException | Fail | InvalidMonthException | month must be between 1 and 12. | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | None | month=1 | month=0 |
| 7.15 | UC-10 | ItemsForMonth_Fail_InvalidMonthHigh | Month > 12 is rejected. | NegativeException | Fail | InvalidMonthException | month must be between 1 and 12. | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | None | month=12 | month=13 |
| 7.16 | UC-10 | ItemsForMonth_Fail_InvalidDayForMonth | If day is provided and invalid for the month (e.g., Feb 31), reject or normalize per contract (here: reject). | NegativeException | Fail | InvalidDayException | day is not valid for month/year. | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | None | e.g., April 31 | day=31, month=4 |
| 7.17 | UC-10 | ItemsForMonth_TimezoneDSTBoundary | Events spanning DST change are included if any overlap is in target month. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | events around DST change dataset | None | DST transition window |
| 7.18 | UC-10 | ItemsForMonth_Edge_EndsExactlyAtStartOfMonth | Event ending exactly at 00:00 on the 1st is excluded (no overlap). | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | end == firstOfMonth at 00:00 | None | exclusive end |
| 7.19 | UC-10 | ItemsForMonth_Edge_StartsExactlyAtEndOfMonth | Event starting exactly at 00:00 of next month is excluded. | Boundary | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | start == firstOfNextMonth at 00:00 | None | exclusive start after month |
| 7.2 | UC-10 | ItemsForMonth_Performance_10kItems | Queries 10,000 items and returns matches efficiently without O(n^2) behavior. | Performance | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | dataset contains 10k items | None | n=10,000 |
| 7.21 | UC-10 | ItemsForMonth_Security_NoPIIInLogs | No sensitive event details are logged during validation errors. | SecurityRef | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | logger redaction | Invalid inputs | N/A |
| 7.22 | UC-10 | ItemsForMonth_Sorting_StableOnTie | Stable sort: when two items start at same instant, original relative order preserved. | Behavior | Pass | None | None | Calendar holds (day, month, year) representing a date context; month ∈ [1,12]; day valid for month/year. ItemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by startDate ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | two items same startTime | None | tie-ordering |
| 7.23 | UC-10 | ItemsForMonth_WhenRepositoryFails | Attempts to fetch all tasks when repository ran into failure. | NegativeException | Fail | ExternalServiceException | Repository failed to perform the operation. | Repository ran into runtime failure. | Repository is online. | Repository is offline. | None |

# CalendarItem

## Test Cases

| Sr. No | Use Case ID | Test Case Name | Test Case Description | Test Case Category | Expected Result (Pass / Fail) | Expected Exception | Expected Exception Message | Preconditions & Invariants | Inputs Partitions (VALID) | Inputs Partitions (INVALID) | Boundary Values |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8.01 | UC-10 | Create_Success_Minimal | Creates an item with valid date, allowed sourceType, and short title. | Positive | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | date=now; sourceType='COURSE'; title='Exam' | None | len(title)=4 |
| 8.02 | UC-10 | Create_Success_TrimmedTitle | Trims leading/trailing spaces in title before storage. | Positive | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | title=' Project kickoff ' -> 'Project kickoff' | None | leading/trailing spaces |
| 8.03 | UC-10 | Create_Success_MaxTitleLength | Accepts title exactly at the 200-char limit. | Boundary | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | len(title)=200 | None | title = 200 |
| 8.04 | UC-10 | Create_Success_UnicodeTitle | Supports Unicode characters (emojis/diacritics) in title. | Positive | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | title='Résumé ✨ – Prüfung' | None | unicode |
| 8.05 | UC-10 | Create_Success_SourceType_CaseInsensitive | Accepts sourceType in any case and stores uppercase. | Positive | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | input 'todo' stored as 'TODO' | None | case normalization |
| 8.06 | UC-10 | Create_Success_FarFutureDate | Allows far-future dates for planning. | Positive | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | date=year 2100 | None | far future |
| 8.07 | UC-10 | Create_Success_FarPastDate | Allows far-past dates for archival imports. | Positive | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | date=year 1970 | None | epoch boundary-ish |
| 8.08 | UC-10 | Create_Fail_NullDate | Null date is rejected. | Negative/Exception | Fail | NullArgumentException | date cannot be null. | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | None | date=null | None |
| 8.09 | UC-10 | Create_Fail_NullSourceType | Null sourceType is rejected. | Negative/Exception | Fail | NullArgumentException | sourceType cannot be null. | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | None | sourceType=null | None |
| 8.1 | UC-10 | Create_Fail_InvalidSourceType | sourceType outside the allowed set is rejected. | Negative/Exception | Fail | InvalidSourceTypeException | sourceType must be one of {COURSE, TODO, SYSTEM, EXTERNAL}. | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | None | sourceType='REMINDER' | None |
| 8.11 | UC-10 | Create_Fail_BlankTitle | Blank or whitespace-only title is rejected. | Negative/Exception | Fail | InvalidTitleException | title cannot be blank. | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | None | title='', ' ' | len=0 |
| 8.12 | UC-10 | Create_Fail_OverMaxTitleLength | Title exceeding 200 chars is rejected. | Negative/Exception | Fail | InvalidLengthException | title exceeds maximum length (200). | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | None | len(title)>200 | len=201 |
| 8.13 | UC-10 | Create_Boundary_DateAtMidnightUTC | Allows dates exactly at 00:00 UTC (no off-by-one due to zone). | Boundary | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | date=YYYY-MM-01T00:00:00Z | None | midnight UTC |
| 8.14 | UC-10 | Create_Boundary_DSTTransition | Stores date around DST transition without shifting. | Boundary | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | date within DST change window | None | DST window |
| 8.15 | UC-10 | Create_Behavior_UnicodeNormalization_NFC | Optionally normalize Unicode title to NFC to avoid duplicate-appearing strings. | Behavior | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | store 'r\u0301sume\u0301' as 'résumé' | None | unicode compose/decompose |
| 8.16 | UC-10 | Create_Security_NoPIIInLogs_OnFailure | Validation failures do not log raw title or date. | Security/NF | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | logger redaction policy | any invalid input | None |
| 8.17 | UC-10 | Create_Performance_BatchCreate_10k | Create 10,000 items without leaks or quadratic behavior. | Performance | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | n=10,000 valid inputs | None | n=10,000 |
| 8.18 | UC-10 | Create_Behavior_StableOrderingByDateTitle | When later sorted by (date asc, title asc), items with equal date keep stable insertion order. | Behavior | Pass | None | None | date != null (instant in UTC); title non-blank, max 200 chars; sourceType ∈ {COURSE, TODO, SYSTEM, EXTERNAL} (case-insensitive input, stored uppercase); title is trimmed (leading/trailing); Unicode preserved; No I/O/logging of raw fields on validation failures. | two items same date | None | tie ordering stable |

# Part 2 – Integration Test

## Workload Division

1. Talha Ahmed → UC-1 to UC-3, UC-5 to UC-6, UC-9
2. Talha Ahmed → Project Setup (Database, Repository, Deployment, UI)
3. Noah Loke →UC-4, UC-10
4. Dechant Trinh → UC-7 to UC-8

## Integration Scope

**IS-1.** Auth → Datastore (Sign-up / Login) – UC-1, UC-2, SD-UC-1, SD-UC-2

**IS-2.** CourseService → Validation → Datastore (Create Course, Add/Update/Complete Tasks) – UC-3 to UC-6, SD-UC-3 to SD-UC-6

**IS-3.** ToDoListService → Validation → Datastore (Create To-Do List, Add/Update/Complete Tasks) – UC-7 to UC-9, SD-UC-7 to SD-UC-9

**IS-4.** CalendarService → Datastore (Enlist Courses tasks and To-Do list tasks and render on UI) – UC-10, SD-UC-10

## Integration Strategy & Order

Using **Hybrid vertical slices** architecture approach for development with features priority ranked by highest user value as enlisted below.

1. IS-1
2. IS-2
3. IS-3
4. IS-4

## Contracts

### Authentication

#### *Signup API*

**Address:** POST /signup {name, email, password}

#### Status Codes

1. 201 {userID}
2. 409 {error_message: Duplicate Email}
3. 400 {error_message: Invalid name, email, or password}
4. 500 {error_message: Server error}

#### *Login API*

**Address:** POST /login {email, password}

#### Status Codes

1. 200 {userID, token}
2. 401 {error_message: Invalid email or password}
3. 400 {error_message: System ran into unexpected error}
4. 500 {error_message: Server error}

## Course

### *Add Course API*

**Address:** POST /users/{uuid}/courses {code, name}

#### Status Codes

1. 201 {courseID}
2. 409 {error_message: Duplicate email}
3. 400 {error_message: Invalid course code or name}
4. 500 {error_message: Server error}

### *Add Course Task API*

**Address:** POST /courses/{cid}/tasks {name, deadline}

#### Status Codes

1. 201 {taskID}
2. 400 {Invalid task name or deadline}
3. 500 {error_message: Server error}

### *Update Course Task Progress API*

**Address:** PATCH /courses/{cid}/tasks/{tid}/progress {value}

#### Status Codes

1. 200 {status, progress}
2. 409 {error_message: Illegal state}
3. 500 {error_message: Server error}

### *Mark Course Task Complete API*

**Address:** POST /courses/{cid}/tasks/{tid}/markComplete

#### Status Codes

1. 200 {status, progress}
2. 500 {error_message: Server error}

## To-Do List

### *Create To-Do List API*

**Address:** POST /users/{uid}/lists {name}

#### Status Codes

1. 201 {listID}
2. 400 {error_message: Invalid input}
3. 500 {error_message: Server error}

#### *Add a To-Do List Task API*

**Address:** POST /lists/{lid}/tasks {description, deadline}

#### Status Codes

1. 201 {taskID}
2. 400 {error_message: Invalid input}
3. 500 {error_message: Server error}

#### *Mark a To-Do list Task Complete API*

**Address:** POST /lists/{lid}/todo-tasks/{tid}/complete

#### Status Codes

1. 200 {status, progress}
2. 500 {error_message: Server error}

### Calendar

#### *Enlist Tasks in a Month API*

**Address:** GET /users/{uid}/calendar?month={*month*}&year={*year*}

#### Status Codes

1. 200 {items: [{date, title, sourceType}...]}
2. 400 {error_message: Invalid Date range}
3. 500 {error_message: Server error}

## Test Design per Module Integration

### S1: Auth ←→ Datastore

- **Happy**: signup unique email → user + credential rows created; login returns token; **DB** has hashed pw.
- **Negatives**: duplicate email (409); wrong password (401); server error (500).

### S2: CourseService ←→ Validation ←→ Datastore

- **Happy**: add course (unique per user) → course row; add task → task row with progress=0; update progress monotonic; complete sets Completed/100.
- **Negatives**: duplicate course code (409).

### S3: ToDoService ←→ Validation ←→ Datastore

- **Happy**: create list; add list task; complete task.
- **Negatives**: blank list name (400).

### S4: CalendarService ←→ Datastore

- **Happy**: month view returns union of CourseTask + ToDoListTask deadlines; performance ≤3s for 200+ items.
- **Negatives**: month with no items → empty array; out-of-range dates (400).

## Trace to Activity Diagram

### Flow F1: "Finish a course task and see it on calendar"

UC-4 (Add Course Task) → UC-5 (Update Progress) / UC-6 (Complete) → UC-10 (View Calendar).

- Steps: UI → CourseSvc → DB → (CTask state) → CalendarSvc → DB → UI
- Proved by: S2-AddTask-Happy, S2-Complete-Happy, S4-MonthView-Happy.

### Flow F2: "Complete a to-do and verify calendar shows it"

UC-8 (Add To-Do list Task) → UC-9 (Mark a To-Do list task Complete) → UC-10 (View Calendar).

- Steps: UI → ToDoListSvc → DB → (TTask state) → CalendarSvc → DB → UI
- Proved by: S3-AddToDoTask-Happy, S3-Complete-Happy, S4-MonthView-Happy.

## Environments & Tooling

1. Local Environment (Docker: app + DB)
2. DB: Postgres
3. Junit (for script-driven testing)
4. Postman for manual sanity testing

# Test Case Inventory (Success Cases)

| Test Case ID | UseCase ID | Integration Scope | Title | Purpose | Preconditions | Steps | Expected Results |
|---|---|---|---|---|---|---|---|
| 1.01 | UC-1 | IS-1 | SignUp_Success_Basic | Registers a new user with valid name, well-formed email, and strong password. | Email not previously registered. Password policy enforced by PasswordPolicy service. | POST /signup | 201; user+cred in db |
| 1.11 | UC-2 | IS-1 | Login_Success | Logs in with correct credentials. | User exists; stored password hash matches. | POST /login | 200; userID from db |
| 1.19 | UC-3 | IS-2 | AddCourse_Success | Adds a new course with a unique code for the user. | User has zero or more courses; code must be unique per user. | POST /users/{uuid}/courses | 201; courseID, code, title in db |
| 1.28 | UC-7 | IS-3 | CreateToDoList_Success | Creates a to-do list with a non-empty name. | User can own multiple lists; no uniqueness constraint assumed. | POST /users/{uid}/lists | 201; listID and name in db |
| 3.07 | UC-4 | IS-2 | AddCourseTask_Success | Adds a task with valid name, future deadline, and optional description. | Deadline must be >= now + 1min. | POST /courses/{cid}/tasks | 201; taskID, title, description, and deadline in db |
| 4.12 | UC-6 | IS-2 | MarkComplete_Success_FromTodo | From TODO, markComplete sets progress=100 & COMPLETED. | progress ∈ [0,100]; status ∈ {TODO, IN_PROGRESS, COMPLETED}; markComplete() sets status=COMPLETED and progress=100; updateProgress(v): if v==0 => TODO, 0<v<100 => IN_PROGRESS, v==100 => COMPLETED; | POST /courses/{cid}/tasks/{tid}/markComplete | 200; |
| 5.01 | UC-8 | IS-3 | AddTask_Success_Minimal | Short non-empty description; deadline slightly in future. | description: non-blank, <= 500 chars; deadline: >= now (instant-based, TZ-safe). Return: ToDoListTask with normalized (trimmed) description. No uniqueness on description. Null args invalid. | POST /users/{uid}/lists | 201; listID and name in db |
| 6.01 | UC-9 | IS-3 | MarkComplete_Success_FromTODO | When status is TODO, markComplete sets status to COMPLETED. | status ∈ {TODO, IN_PROGRESS, COMPLETED}. markComplete(): sets status=COMPLETED and does not modify description or deadline. Operation is idempotent (calling again keeps status=COMPLETED). No side effects on enclosing list; no I/O; | POST /lists/{lid}/todo-tasks/{tid}/complete | 200; task progress in db |
| 7.03 | UC-10 | IS-4 | ItemsForMonth_Success_MultiDayWithinMonth | Returns multi-day events entirely within the target month. | Calendar holds (day, month, year) representing a date context; month ∈ [1,12], day valid for month/year. itemsForMonth(User) returns all CalendarItem instances that intersect the given month for that user (events starting/ending within month or spanning across). User must be non-null; only items owned by or shared with the user are returned. Results are sorted by start date ascending (stable), timezone-agnostic (UTC instants). If (day) is out-of-range for given month/year, method validates month/year and ignores day for filtering. | GET /users/{uid}/calendar?month={*month* }&year={*year* } | 200; list of tasks |