

Example M5: Testing and Code Review

1. Change History

Change Date	Modified Sections	Rationale
2025-03-31	2.1.1, 2.1.2, 2.3, 2.4	Added more tests for sessionRecommender, so updated links and coverage screenshots

2. Back-end Test Specification: APIs

2.1. Locations of Back-end Tests and Instructions to Run Them

2.1.1. Tests

Interface	Describe Group Location, No Mocks	Describe Group Location, With Mocks	Mocked Components
GET /auth/verify	Link to unmocked test	Link to mocked test	User table
POST /auth/google	Link to unmocked test	Link to mocked test	User table
PUT /auth/profile/:googleId	Link to unmocked test	Link to mocked test	User table
POST /group	Link to unmocked test	Link to mocked test	User table
GET /group/:userId	Link to unmocked test	Link to mocked test	Group table
PUT /group/:groupId	Link to unmocked test	Link to mocked test	Group table
DELETE /group/:groupId	Link to unmocked test	Link to mocked test	Group table
POST /notification/deviceToken	Link to unmocked test	Link to mocked test	Device table
DELETE /notification/deviceToken	Link to unmocked test	Link to mocked test	Device table
PUT /session/:sessionId/join	Link to unmocked test	Link to mocked test	Session table

Interface	Describe Group Location, No Mocks	Describe Group Location, With Mocks	Mocked Components
POST /session	Link to unmocked test	Link to mocked test	User table
PUT /session/:sessionId/leave	Link to unmocked test	Link to mocked test	Session table
DELETE /session/:sessionId	Link to unmocked test	Link to mocked test	Session table
GET /session/availableSessions/:userId	Link to unmocked test	Link to mocked test	Session table
GET /session/nearbySessions/	Link to unmocked test	Link to mocked test	Session table
PUT /user/friendRequest	Link to unmocked test	Link to mocked test	User table
GET /user/friendRequests	Link to unmocked test	Link to mocked test	User table
GET /user/friends	Link to unmocked test	Link to mocked test	User table
PUT /user/friend	Link to unmocked test	Link to mocked test	User table
DELETE /user/removeFriend	Link to unmocked test	Link to mocked test	User table
GET /user	Link to unmocked test	Link to mocked test	User table
PUT /user	Link to unmocked test	Link to mocked test	User table
DELETE /user	Link to unmocked test	Link to mocked test	User table
sendPushNotification() helper function	N/A	Link to mocked test	Device table, Firebase Notification API
sessionRecommender helper function	Link to unmocked test	N/A	N/A

2.1.2. Commit Hash Where Tests Run

[a907008f2fb77dc20720091d11a77821406f0f0c](#)

2.1.3. Explanation on How to Run the Tests

1. Clone the Repository:

- Open your terminal and run:

```
git clone https://github.com/mayankrastogi02/cpen321-study-wimme.git
```

2. Install node packages

- Navigate to the **server** folder in your cloned repository and run:

```
npm i
```

3. Run jest tests with coverage

- Run the Jest tests with coverage using the following command:

```
npm test --coverage
```

4. Create a .env file

- Make a .env file under the **server** folder and add the following keys

```
DB_URI=<YOUR MONGODB URI>  
PORT=<YOUR PORT>  
GCP_PROJECT_ID=study-wimme
```

5. View coverage report

- Navigate to **<YOUR REPO>/server/coverage/lcov-report** to view the html coverage reports in a browser

2.2. GitHub Actions Configuration Location

<YOUR REPO>/github/workflows/test-backend.yml

2.3. Jest Coverage Report Screenshots With Mocks

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	93.71	97.42	86.27	93.77	
server	36.17	62.5	22.22	36.17	
index.ts	36.17	62.5	22.22	36.17	28-34,71-72,79-90,96-103,108-144
server/controllers	100	98.96	100	100	
AuthController.ts	100	93.93	100	100	70
GroupController.ts	100	100	100	100	
NotificationController.ts	100	100	100	100	
SessionController.ts	100	100	100	100	
UserController.ts	100	100	100	100	
server/routes	100	100	100	100	
AuthRoutes.ts	100	100	100	100	
GroupRoutes.ts	100	100	100	100	
NotificationRoutes.ts	100	100	100	100	
SessionRoutes.ts	100	100	100	100	
UserRoutes.ts	100	100	100	100	
server/schemas	100	100	100	100	
DeviceSchema.ts	100	100	100	100	
GroupSchema.ts	100	100	100	100	
SessionSchema.ts	100	100	100	100	
UserSchema.ts	100	100	100	100	
server/utils	98.18	96.77	100	100	
notificationUtils.ts	100	100	100	100	
sessionRecommender.ts	97.14	95.45	100	100	54
Test Suites: 10 passed, 10 total Tests: 136 passed, 136 total Snapshots: 0 total Time: 9.929 s, estimated 10 s					

We were able to achieve a **93.7%** statement coverage and a **97.4%** branch coverage when we ran our report with mocks

- Our index.ts file did not have a lot of coverage. This was intentional because we didn't need we want to use an actual database for testing, instead using an in-memory database. Additionally, we did not need firebase credentials to run the test. Thus we added a boolean condition to exclude this code during testing. We also did not want to run the cron timing jobs when running the tests as those run continuously and do not stop, hence all those lines are uncovered as well.
- We also had 1 unreachable line of code in AuthController which we are going to modify for the final milestone.

2.4. Jest Coverage Report Screenshots Without Mocks

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	83.56	94.84	84.31	83.6	
server	36.17	62.5	22.22	36.17	
index.ts	36.17	62.5	22.22	36.17	28-34, 71-72, 79-90, 96-103, 108-144
server/controllers	88.88	98.96	100	88.7	
AuthController.ts	92.15	93.93	100	91.11	44-45, 149-150
GroupController.ts	88.88	100	100	88.88	63-64, 106-107, 129-130
NotificationController.ts	86.95	100	100	86.95	40-41, 56
SessionController.ts	89.09	100	100	89.09	76-77, 107-108, 164-165, 211-212, 271-272, 326-327
UserController.ts	87.78	100	100	87.78	58-59, 87-88, 149-150, 178-179, 227-228, 251-252, 294-295, 342-343
server/routes	100	100	100	100	
AuthRoutes.ts	100	100	100	100	
GroupRoutes.ts	100	100	100	100	
NotificationRoutes.ts	100	100	100	100	
SessionRoutes.ts	100	100	100	100	
UserRoutes.ts	100	100	100	100	
server/schemas	100	100	100	100	
DeviceSchema.ts	100	100	100	100	
GroupSchema.ts	100	100	100	100	
SessionSchema.ts	100	100	100	100	
UserSchema.ts	100	100	100	100	
server/Utils	81.81	77.41	91.66	84	
notificationUtils.ts	55	33.33	66.66	57.89	18-49
sessionRecommender.ts	97.14	95.45	100	100	54

We saw a slight decline in both statement and branch coverage when we ran our tests without mocks at **83.6%** and **94.8%** respectively

- This was partially attributed to the fact that all of our endpoints were wrapped in a try-catch block that threw an internal server error with code 500 if the server were to unexpectedly fail (ie. from a database error). We were unable to test these unexpected failures without mocks.
- Additionally, the push notification API for firebase admin were unable to be tested without because we could not generate real valid/expired device tokens, hence the low coverage in `notificationUtils.ts`.

3. Back-end Test Specification: Tests of Non-Functional Requirements

3.1. Test Locations in Git

Non-Functional Requirement	Location in Git
Performance (Response Time)	app\app\src\androidTest\java\com\cpen321\study_wimme\Non-Functional Tests\NFR3_Tests.kt
Reliability and Error Handling	app\app\src\androidTest\java\com\cpen321\study_wimme\Non-Functional Tests\NFR4_Tests.kt

3.2. Test Verification and Logs

- **Performance (Response Time)**
 - **Verification:** The response time of the API is measured using the `NFR3_Tests.kt` test suite. The test suite sends multiple requests to the API and measures the time taken for the API to respond. The response time is then compared against the expected response time to verify that the API meets the performance requirements. As mentioned in our requirements, the API should respond within 300ms for 95% of the requests. The test reports the response times for each request, the

average response time, as well as the maximum response time. If the average response time is less than 300ms and the maximum response time is less than 300ms, the test is considered successful.

- **Log Output**

```
...
Average Response Time: 129.26315789473685 ms
Max Response Time: 213 ms
Fastest Response Time: 77 ms
run finished: 23 tests, 0 failed, 0 ignored
...
```

- **Reliability and Error Handling**

- **Verification:** The reliability and error handling of the API are tested using the `NFR4_Tests.kt` test suite. The test suite sends requests to the API with different inputs to test the API's error handling capabilities. The test suite verifies that the API returns the correct error codes and messages when invalid inputs are provided. The test suite also verifies that the API returns the correct error codes and messages when the API encounters internal errors. The test suite reports the error codes and messages returned by the API for each request and compares them against the expected error codes and messages to verify that the API meets the reliability and error handling requirements. If the API returns the correct error codes and messages for all requests, the test is considered successful.

- **Log Output**

```
started:
testHostSessionMissingFields(com.cpen321.study_wimme.ErrorRecoveryTests
)
finished:
testHostSessionMissingFields(com.cpen321.study_wimme.ErrorRecoveryTests
)
started:
testCreateOrUpdateUserMissingFields(com.cpen321.study_wimme.ErrorRecover
yTests)
finished:
testCreateOrUpdateUserMissingFields(com.cpen321.study_wimme.ErrorRecover
yTests)
started:
testDeleteSessionMissingSessionId(com.cpen321.study_wimme.ErrorRecovery
Tests)
finished:
testDeleteSessionMissingSessionId(com.cpen321.study_wimme.ErrorRecovery
Tests)
started:
testVerifyUserMissingGoogleId(com.cpen321.study_wimme.ErrorRecoveryTest
s)
finished:
testVerifyUserMissingGoogleId(com.cpen321.study_wimme.ErrorRecoveryTest
s)
```

```

    started:
testCreateGroupMissingFields(com.cpen321.study_wimme.ErrorRecoveryTests
)
    finished:
testCreateGroupMissingFields(com.cpen321.study_wimme.ErrorRecoveryTests
)
    started:
testGetGroupsMissingUserId(com.cpen321.study_wimme.ErrorRecoveryTests)
    finished:
testGetGroupsMissingUserId(com.cpen321.study_wimme.ErrorRecoveryTests)
    started:
testUpdateUserProfileMissingGoogleId(com.cpen321.study_wimme.ErrorRecov
eryTests)
    finished:
testUpdateUserProfileMissingGoogleId(com.cpen321.study_wimme.ErrorRecov
eryTests)
    started:
testAssociateDeviceMissingFields(com.cpen321.study_wimme.ErrorRecoveryT
ests)
    finished:
testAssociateDeviceMissingFields(com.cpen321.study_wimme.ErrorRecoveryT
ests)
run finished: 8 tests, 0 failed, 0 ignored

```

4. Front-end Test Specification

4.1. Location in Git of Front-end Test Suite:

```

cpen321-study-
wimme\app\app\src\androidTest\java\com\cpen321\study_wimme\EspressoTests.kt

```

4.2. Tests

- **Use Case: Manage session**
 - **Test Case: Create session**
 - **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The actor clicks on the create session button	Open activity and click create session button.
2. The system displays empty, editable fields for time range, location, and description. It also includes a toggle that lets the actor choose if the session is private or public If the session is private, the actor chooses which friends or groups they broadcast the session to	Implicitly checks visibility of elements (using findViewById, withId), toggles session to be public.

Scenario Steps	Test Case Steps
3a. User enters invalid information for session (letters for time, symbols for anything, date in the past, etc...)	Input an empty string for the session name field.
3a1. Message informing user that they have entered invalid information for a field	Click on field with error image (session name field), checking that message "Session name is required" is displayed.
3. The actor clicks the fields, enters the appropriate information and specifies whether the session is public or private. If the session is private, the actor chooses which friends or groups they broadcast the session to	Clicks on fields and fills out session details with valid information.
4. The actor clicks the create button	Click host session button.
5. The inputted data gets populated in the database for the new session	Check that hosted session is visible in study list.
6. The system displays that the session has been created successfully. If the session is private, the selected friends/groups are notified.	Check that hosted session is visible in study list. Visual verification of Toast message.

- **Use Case: Join + Leave session (tested together)**

- **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The actor clicks on the session which they want to join	Navigate to public sessions in session list and click on a session.
1a. User already joined	Click into session and click join session again after initial join.
1a1. System displays message that user is already in session and returns to session list page	Check return to session list page and visually verify toast message.
2. The system retrieves the session information from the database and displays the information	Checks that session details activity has started by selecting the join button that exists in that activity.
3. The actor clicks the join button	Click join session button.
4. The system updates the database to include the actor as an attendee	Check that session appears in the joined sessions list.
5. The system displays that the actor has joined the session	Visual verification of toast message.

Scenario Steps	Test Case Steps
Leave Session	
1. The actor clicks on the session which they have joined	Navigate to "joined" sessions in the session list and click on previously joined session.
2. The system retrieves the session information from the database and displays the information	Check that the session details activity has started by verifying the "Leave" button is present and clickable in that activity.
3. The actor clicks the leave button	Click the "Leave Session" button.
3a. Session no longer exists	Requires server mocks so untested.
3a1. System displays error that session cannot be found and returns to sessions list	Requires server mocks so untested.
4. The system updates the database to remove the actor as an attendee	Check that session no longer appears in the joined sessions list.
5. The system displays that the actor has left the session	Visual verification of toast message.

- **Use Case: Manage session**

- **Test Case: Delete session**
- **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The actor clicks the session which they want to delete	Navigate to hosted sessions in session list and click on a self-hosted session.
2.The system retrieves the session information from the database and displays the information	Checks that session details activity has started by selecting the delete button that exists in that activity
3. The actor clicks the delete button	Click delete session button.
4. The system displays a popup asking to confirm deletion of this session	Check that popup is shown to user by selecting the cancel button that is only visble in popup.
4a. User cancels deletion	Cancel button on popup is clicked.
4a1. System closes confirmation popup and returns to session page with no changes.	Return to session detail pages and checks by selecting delete button again.
5. The actor clicks the confirm deletion button	Delete button is clicked again and confirm is clicked on popup.

Scenario Steps	Test Case Steps
6. The system deletes the session entry from the database	Check that hosted session is no longer visible in study list.
7. The system displays that the session has been deleted successfully	Check that hosted session is not visible in study list. Visual verification of Toast message.

- **Use Case: Manage profile**

- **Test Case: Create Profile**

- **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The actor clicks the create profile button.	Start activity session with complete_profile set to true (creating profile for the first time).
2. The system displays empty, editable fields about their profile.	Implicitly verifies that expected fields exist through selection.
3. The actor clicks the fields and enters the appropriate information	Fill out fields with valid inputs.
3a. User enters invalid text for respective fields (string for year, number for major, etc..)	Enter empty string for name.
3a1. Message is displayed to user informing them that they have entered invalid characters for the given field(s) and to fix it before submitting is allowed.	Verify expected error shows up.
4. The actor clicks the save button	Click save button.
5. The inputted data gets populated in the database for that user.	Close and open profile again to verify entered fields were properly saved.
6. The system displays that the profile has been created successfully	Visual verification of displayed toast message.

- **Use Case: Manage profile**

- **Test Case: Read Profile**

- **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The actor clicks on their profile button	Click on profile button.

Scenario Steps	Test Case Steps
2. The system retrieves the actor's profile information from the database	Verifies that expected fields (from creation) are shown correctly (since they come from the database)
3. The system displays information about their profile: username, year, and faculty	Verifies that expected fields (from creation) are shown correctly with expected values

- **Use Case: Manage profile**

- **Test Case: Edit Profile**
- **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The actor clicks the edit button on their profile	Start activity session with complete_profile set to false (profile already exists).
2. The system retrieves the actor's profile information from the database	Verifies that expected fields are not empty (since they come from the database)
3. The system displays information about their profile, such as username, year, and faculty as editable fields	Verifies that expected fields are not empty
4. The actor clicks on the field they want to modify and changes it	New information is entered for each field
4a. User enters invalid text for respective fields (string for year, number for major, etc..)	Enter empty string for name.
4a1. Message is displayed to user informing them that they have entered invalid characters for the given field(s) and to fix it before submitting is allowed.	Verify expected error shows up.
5. The actor clicks the save button	Close and open profile again to verify entered fields were properly saved.
6. The inputted data gets updated in the database for that user	Visual verification of displayed toast message.
7. The system displays that the changes have been saved successfully	Visual verification of displayed toast message.

◦ **Test Logs:**

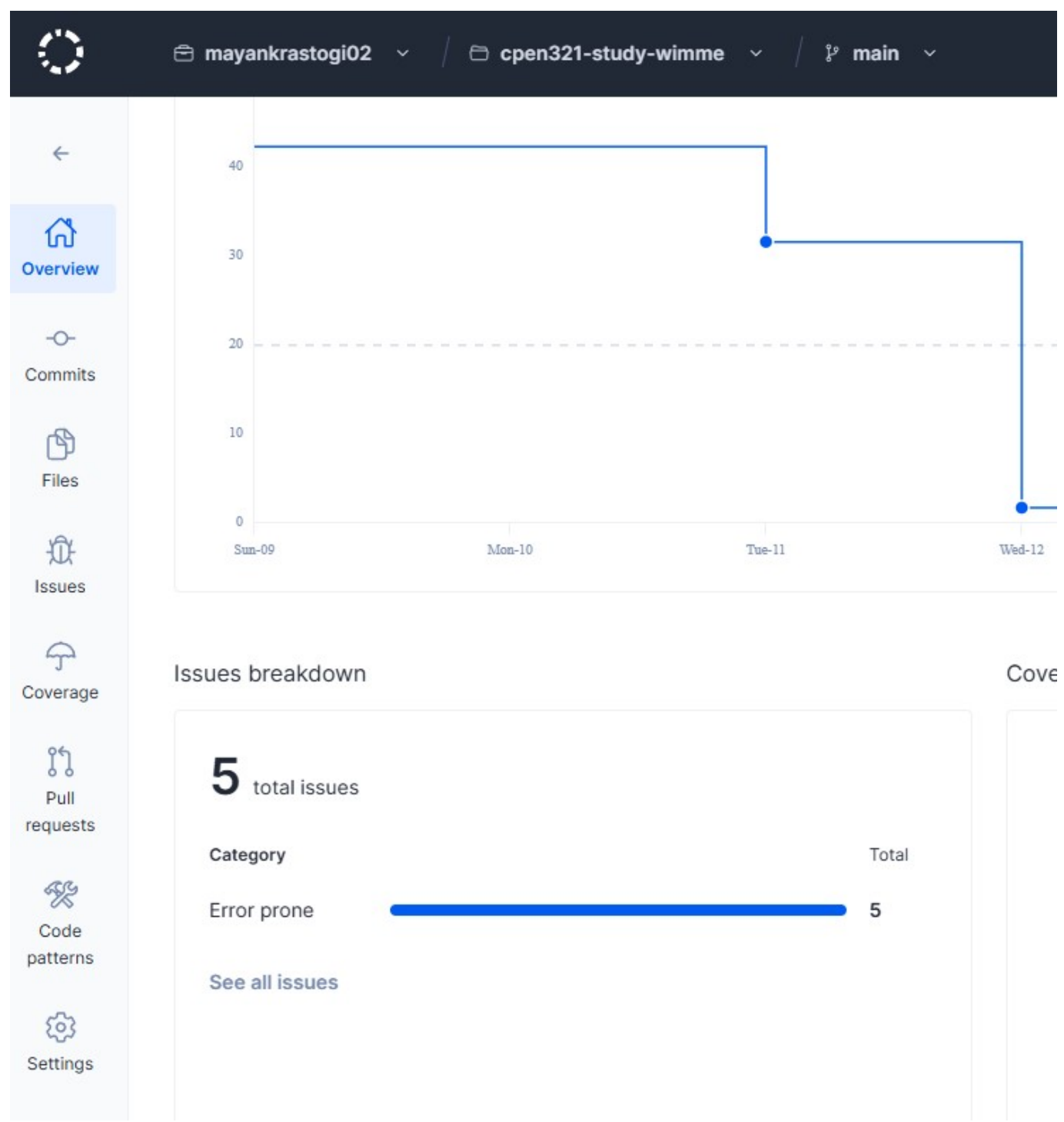
✓ Test Results	1m	6/6	> Task :app:connectedDebugAndroidTest
✓ CreateSessionActivityTest	1m	6/6	Starting 6 tests on Copy_of_Copy_of_Pixel_9_API_31(AVD) - 12
✓ test01_CreateSessionWithValidInputs	13s	✓	Connected to process 19174 on device 'Copy_of_Copy_of_Pixel_9_API_31 [emulator-5554]'.
✓ test02_JoinAndLeaveSession	16s	✓	Copy_of_Copy_of_Pixel_9_API_31(AVD) - 12 Tests 0/6 completed. (0 skipped) (0 failed)
✓ test03_DeleteHostedSession	7s	✓	Copy_of_Copy_of_Pixel_9_API_31(AVD) - 12 Tests 1/6 completed. (0 skipped) (0 failed)
✓ test04_NewUserCompletingProfilePrepopulate	7s	✓	Copy_of_Copy_of_Pixel_9_API_31(AVD) - 12 Tests 3/6 completed. (0 skipped) (0 failed)
✓ test05_ReadProfileFields	1s	✓	Copy_of_Copy_of_Pixel_9_API_31(AVD) - 12 Tests 5/6 completed. (0 skipped) (0 failed)
✓ test06_EditProfileFieldsAndSave	14s	✓	Finished 6 tests on Copy_of_Copy_of_Pixel_9_API_31(AVD) - 12
			BUILD SUCCESSFUL in 1m 18s
			68 actionable tasks: 5 executed, 63 up-to-date

5. Automated Code Review Results

5.1. Commit Hash Where Codacy Ran

d6acd47aac4455d32b4aa4fb980f108619e182ae

5.2. Unfixed Issues per Codacy Category



5.3. Unfixed Issues per Codacy Code Pattern

All issues 5

Code patterns

Too many functions inside a/an file/class/... 4

Others 1

Class 'HomeFragment' with '18' functions detected. Defined threshold inside classes is set to '11'

app/app/src/main/java/com/cpen321/study_wimme/HomeFragment.kt

```
33 class HomeFragment : Fragment() {
```

Class 'UserSettingsActivity' with '12' functions detected. Defined threshold inside classes is set to '11'

app/app/src/main/java/com/cpen321/study_wimme/UserSettingsActivity.kt

```
26 class UserSettingsActivity : AppCompatActivity() {
```

Class 'CreateSessionActivity' with '11' functions detected. Defined threshold inside classes is set to '11'

app/app/src/main/java/com/cpen321/study_wimme/CreateSessionActivity.kt

```
32 class CreateSessionActivity : AppCompatActivity() {
```

Class 'FriendsFragment' with '11' functions detected. Defined threshold inside classes is set to '11'

app/app/src/main/java/com/cpen321/study_wimme/FriendsFragment.kt

```
32 class FriendsFragment : Fragment() {
```

allowed, because there is no way to get the instance of an outer class from an inner class in Kotlin.

```
val range = listOf<String>("foo", "bar")
loop@ for (r in range) {
    if (r == "bar") break@loop
    println(r)
}

class Outer {
    inner class Inner {
        fun f() {
            val i = this@Inner // referencing itself, use `this` instead
        }
    }
}
```

Related code pattern: [Expression with labels increase complexity and affect maintainab](#) by det... Applied from configuration file

Expression with labels increase complexity and affect maintainability.

app/app/src/main/java/com/cpen321/study_wimme/CreateSessionActivity.kt

```
200 return@setOnClickListener
```

5.4. Justifications for Unfixed Issues

- Code Pattern: Too many functions inside a/an file/class/object/interface always indicate a violation of the single responsibility principle. Maybe the file/class/object/interface wants to manage too many things at once.**

1. Class 'HomeFragment' with '18' functions detected. Defined threshold inside classes is set to '11'

Location in Git:

app/app/src/main/java/com/cpen321/study_wimme/HomeFragment.kt

- Justification:** The `HomeFragment` class contains multiple functions that handle different responsibilities, such as fetching user data, displaying user data, and handling user interactions. Attempts were made to refactor the code to suppress the `method too long` warning, but as a result, the class now contains multiple functions that handle different responsibilities. This is a tradeoff between code readability and adherence to the single responsibility principle.

2. **Class 'UserSettingsActivity' with '12' functions detected. Defined threshold inside classes is set to '11'**
 - **Location in Git:**
app/app/src/main/java/com/cpen321/study_wimme/UserSettingsActivity.kt
 - **Justification:** The `UserSettingsActivity` class contains multiple functions that handle different responsibilities, such as updating user settings, displaying user settings, and handling user interactions. Attempts were made to refactor the code to suppress the `method too long` warning, but as a result, the class now contains multiple functions that handle different responsibilities. This is a tradeoff between code readability and adherence to the single responsibility principle.
 3. **Class 'CreateSessionActivity' with '11' functions detected. Defined threshold inside classes is set to '11'**
 - **Location in Git:**
app/app/src/main/java/com/cpen321/study_wimme/CreateSessionActivity.kt
 - **Justification:** The `CreateSessionActivity` class contains multiple functions that handle different responsibilities, such as creating a new study session, displaying study session details, and handling user interactions. Attempts were made to refactor the code to suppress the `method too long` warning, but as a result, the class now contains multiple functions that handle different responsibilities. This is a tradeoff between code readability and adherence to the single responsibility principle.
 4. **Class 'FriendsFragment' with '11' functions detected. Defined threshold inside classes is set to '11'**
 - **Location in Git:**
app/app/src/main/java/com/cpen321/study_wimme/FriendsFragment.kt
 - **Justification:** The `FriendsFragment` class contains multiple functions that handle different responsibilities, such as fetching user data, displaying user data, and handling user interactions. Attempts were made to refactor the code to suppress the `method too long` warning, but as a result, the class now contains multiple functions that handle different responsibilities. This is a tradeoff between code readability and adherence to the single responsibility principle.
- **Others**
 1. **Expression with labels increase complexity and affect maintainability. - - Location in Git:**
app/app/src/main/java/com/cpen321/study_wimme/ CreateSessionActivity.kt -
Justification: The `CreateSessionActivity` class uses labeled expressions to manage complex control flows that are essential for the application's functionality. Removing these labels would significantly increase the complexity of the code and reduce its maintainability. Therefore, the decision was made to keep the labeled expressions to ensure the code remains understandable and maintainable.