Testing and Code Review

1. Change History

Change Date	Modified Sections	Rationale
2025-02-14	Section 2 (Back-end Test Specification)	Documented Jest-based API testing setup and execution flow.
2025-02-15	Section 2 (Back-end Test Specification)	Added unit-test coverage details for auth layer.
2025-02-15	Section 2 (Back-end Test Specification)	Rebuilt auth test suites with mocked/unmocked coverage.

---

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	Descr
Auth Routes	`backend/tests/unmock/auth/auth.routes.spec.ts`	`backend/tests`
Auth Controller	`backend/tests/unmock/auth/auth.controller.spec.ts`	`backend/tests`
Auth Middleware	`backend/tests/unmock/auth/auth.middleware.spec.ts`	`backend/tests`

Auth Service	`backend/tests/unmock/auth/auth.service.spec.ts`	`backend/tests`

2.1.2. Commit Hash Where Tests Run

The first successful run with the current suite has not yet been recorded. After executing the tests locally (requires MongoDB binaries for `mongodb-memory-server`), capture the SHA via `git rev-parse HEAD` and record it here.

2.1.3. Explanation on How to Run the Tests

1. **Install dependencies**

```
cd backend
npm install
```

2. **Run all tests**

```
npm run test
```

   - The suites are split between `tests/unmock/**` (no dependency mocks) and `tests/mocked/**` (external collaborators mocked per requirement).
   - Firebase Admin access is stubbed via local jest mocks; no service account file is required.
   - Some environments block binding ephemeral ports; if you see `listen EPERM` locally, rerun with `SKIP_MONGO=true` to disable in-memory MongoDB hooks.

3. **Watch mode (optional)**

```
npm run test:watch
```

4. **Generate coverage**

```
npm run test:coverage
```

2.2. Jest Configuration Location

`backend/jest.config.ts`

> *Note:* CI automation for backend tests is not yet committed. When a GitHub Actions workflow is added, document its path here as well.

2.3. Jest Coverage Report Screenshots for Tests Without Mocking

(Placeholder for Jest coverage screenshot without mocking)

2.4. Jest Coverage Report Screenshots for Tests With Mocking

(Placeholder for Jest coverage screenshot with mocking)

2.5. Jest Coverage Report Screenshots for Both Tests With and Without Mocking

(Placeholder for Jest coverage screenshot both with and without mocking)

---

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

3.1. Test Locations in Git

Non-Functional Requirement	Location in Git
Recognition Latency (≤ 10 s end-to-end)	`backend/tests/nonfunctional/recognitionLatency.spec.ts`
Privacy & Data Protection (Account Deletion)	`backend/tests/nonfunctional/privacyDeletion.spec.ts`

3.2. Test Verification and Logs

- **Recognition Latency (≤ 10 s end-to-end)**
  - **Verification:** A Jest + Supertest suite hits `POST /api/recognition`, injects canned Zyla responses via `jest.spyOn(recognitionService, 'recognizeFromUrl')`, and

measures the elapsed wall-clock time between request dispatch and JSON payload delivery. The test iterates through three representative payload descriptors (1 MB, 3 MB, 5 MB) and fails if any response exceeds the 10 000 ms SLA, giving us a fast regression signal that controller/middleware changes didn't bloat latency. Execute with `npm run test:nfr-recognition`.

- **Log Output**

```
$ cd backend && npm run test:nfr-recognition
PASS tests/nonfunctional/recognitionLatency.spec.ts
  NFR: Recognition latency
    ✓ { label: '1MB payload', body: [Object] } completes within 10 second
    ✓ { label: '3MB payload', body: [Object] } completes within 10 second
    ✓ { label: '5MB payload', body: [Object] } completes within 10 second
```

- **Privacy & Data Protection (Account Deletion)**
  - **Verification:** Using `mongodb-memory-server`, the suite provisions a user, catalogs, and friendships, then invokes `DELETE /api/user/profile`. It verifies that every protected route returns 401 without a JWT, 200 with a valid token, and that after deletion the `users`, `catalogs`, `entries`, and `friendships` collections contain no documents tied to the deleted user. The test also confirms that subsequent authorized requests fail with 401, proving the token is invalidated. Execute via `npm run test:nfr-privacy`.
  - **Log Output**

```
$ cd backend && npm run test:nfr-privacy
PASS tests/nonfunctional/privacyDeletion.spec.ts
  NFR: Privacy & Data Protection
    ✓ catalog endpoints reject unauthenticated access (153 ms)
    ✓ deleting a profile removes personal data and invalidates the token
```

---

4. Front-end Test Specification

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

`frontend/src/androidTest/java/com/studygroupfinder/`

4.2. Tests

- **Use Case: Login**

- **Expected Behaviors:**

Scenario Steps	Test Case Steps
1. The user opens "Add Todo Items" screen.	Open "Add Todo Items" screen.
2. The app shows an input text field and an "Add" button. The add button is disabled.	Check that the text field is present on screen. Check that the button labelled "Add" is present on screen. Check that the "Add" button is disabled.
3a. The user inputs an ill-formatted string.	Input "_^_^^OQ#$" in the text field.
3a1. The app displays an error message prompting the user for the expected format.	Check that a dialog is opened with the text: "Please use only alphanumeric characters ".
3. The user inputs a new item for the list and the add button becomes enabled.	Input "buy milk" in the text field. Check that the button labelled "add" is enabled.
4. The user presses the "Add" button.	Click the button labelled "add ".
5. The screen refreshes and the new item is at the bottom of the todo list.	Check that a text box with the text "buy milk" is present on screen. Input "buy chocolate" in the text field. Click the button labelled "add". Check that two text boxes are present on the screen with "buy milk" on top and "buy chocolate" at the bottom.
5a. The list exceeds the maximum todo-list size.	Repeat steps 3 to 5 ten times. Check that a dialog is opened with the text: "You have too many items, try completing one first".

- **Test Logs:**

```
[Placeholder for Espresso test execution logs]
```

- Use Case: ...
  - Expected Behaviors:

Scenario Steps	Test Case Steps
...	...

  - Test Logs:

    ```
    [Placeholder for Espresso test execution logs]
    ```

- ...

---

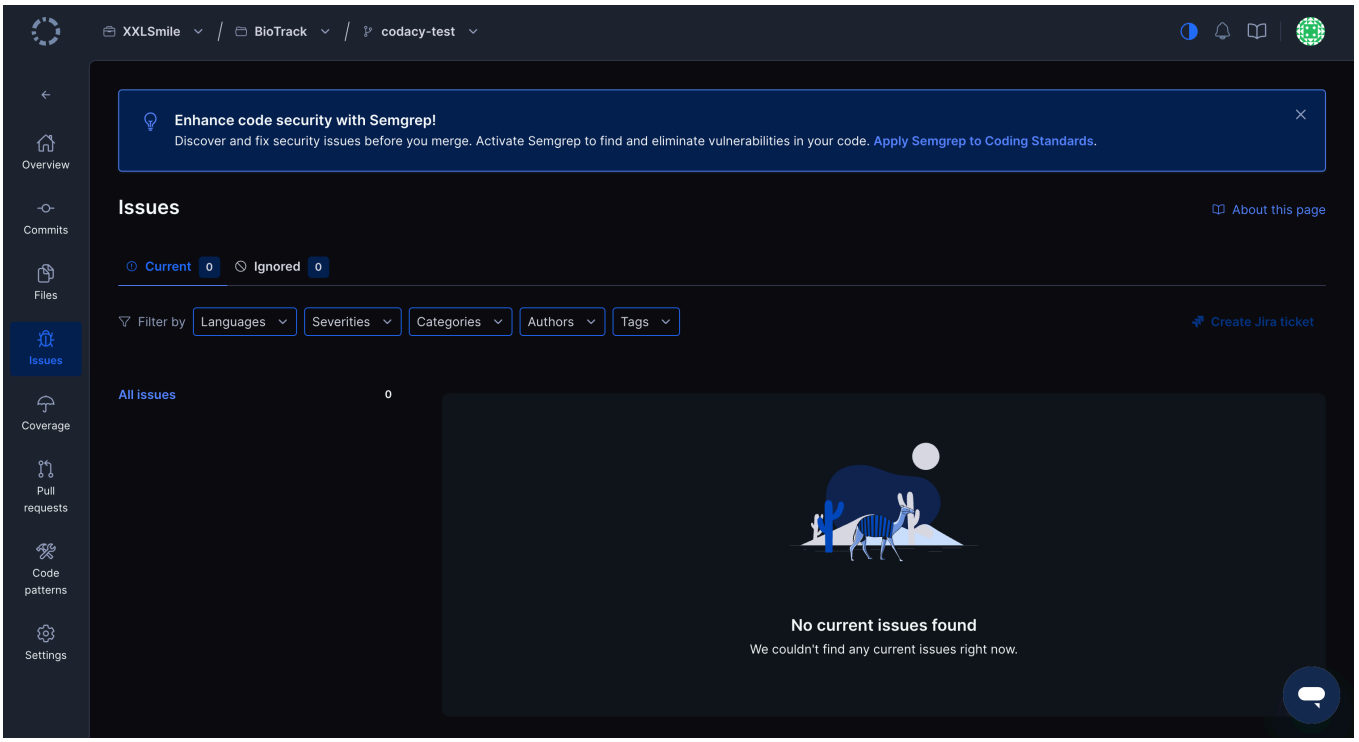
5. Automated Code Review Results

5.1. Commit Hash Where Codacy Ran

`527afd4`

5.2. Unfixed Issues per Codacy Category

5.3. Unfixed Issues per Codacy Code Pattern



5.4. Justifications for Unfixed Issues

- None