Code and Test Review

1. Manual Code Review

- The code is well-structured and readable but lacks robust validation and error handling, which impacts its security and reliability.
- **Score: 8.5/10**

2. Manual Test Review

- **a. Completeness:**
    - All APIs exposed to the frontend are tested.
    - Three main use cases are tested.
    - Errors and edge cases are thoroughly tested.
    - Correct assertions are used.
    - **Score: 10/10**
    - **Reasoning:**
      The tests are strong and well-structured, with excellent coverage of the three main use cases, edge cases and assertions.
- **b. Requirement & Design Match:** Tests match the requirements and design.
    - **Score: 10/10**
    - **Reasoning:**
      The provided tests comprehensively cover the functional and non-functional requirements, design specifications, and main project complexity
- **c. Structure:** Tests are well-structured.
    - **Score: 10/10**
- **d. Coverage:** Tests provide good coverage.
    - **Score: 8/10**
    - **Reasoning:**
      The overall coverage is decent, the tests could use some additional coverage for RAGService.ts, studyService.ts, and index.ts
- **e. Non-Functional Requirements:** Non-functional requirements (e.g., performance, security) are tested well.
    - **Score: 9/10**
    - **Reasoning:**
      The tests are well-structured and enforce essential key performance criteria, but improvements in scalability, and failure handling would make them more robust. Even though this test fails due to this being in the MVP stage, the test adequately shows that this is not ready, as would be expected from the test. This does not mean that the test is bad, instead that the code is inadequate which is out of the scope of this grade

- **f. Automation:** All backend tests can be run automatically.
    - **Score: 10/10**
    - **Reasoning:**
      All backend tests are well-documented and run with very few terminal prompts. They are all automatic, easy to run, and show all important information about the testing. All used endpoints are tested.

3. Automated Code Review

- **a. Codacy Setup:** Codacy runs with the required setup.
    - **Score: 10/10**
- **b. Codacy Issues:** All remaining Codacy issues are well-justified.
    - **Score: 10/10**

Justifications: The Codacy configuration files were properly setup and the justifications for all remaining warnings were more than reasonable, and had adequate citations and justifications.

4. Fault Report

There is a severe vulnerability in the `DocumentUploadScreen` that arises from the lack of proper file validation, which could allow users to upload malicious files disguised as supported formats (e.g., PDF, JPEG). Without verifying the file's actual content or type, the application might accept harmful files, such as scripts or executables, leading to potential security breaches. This could enable attackers to exploit backend systems, compromise user data, or execute unauthorized code. This exists due to the complete absence of frontend and backend validation and creates a significant risk, as malicious files could bypass initial checks and cause damage during processing or storage. This issue highlights the importance of thorough file validation and content inspection to prevent security threats.