1. UserQuestionCountAPITest:
   * Purpose: This test case validates the user\_question\_count\_view API, which retrieves the count of total favorite and read questions per user, paginated to 100 users per page.
   * Test Scenario: The test sets up two user profiles and their associated favorite and read questions. It then makes a GET request to the API and checks if the response contains the correct user information and their respective favorite and read question counts.
   * Test Explanation: The test verifies that the API returns a status code of 200 (OK) and the correct user information with their favorite and read question counts.
2. FilteredQuestionViewAPITest:
   * Purpose: This test case validates the filtered\_question\_view API, which filters questions based on their read, unread, favorite, and unfavorite status.
   * Test Scenario: The test sets up test data with a read and a favorite question. It then makes a GET request to the API with the status parameter set to 'read' to retrieve read questions. The test checks if the response contains the correct read question based on the status parameter.
   * Test Explanation: The test verifies that the API returns a status code of 200 (OK) and the correct read question based on the 'read' status parameter.
3. AddReadQuestionAPITest:
   * Purpose: This test case validates the add\_read\_question API, which allows adding read questions for a user.
   * Test Scenario:
     + Test 1: The test makes a valid POST request with a valid payload containing a valid user\_id and question\_id. It checks if the API returns a status code of 201 (CREATED) and the read question is added to the database.
     + Test 2: The test makes an invalid POST request with an invalid user\_id (non-existing user). It checks if the API returns a status code of 400 (BAD REQUEST) and the read question is not added to the database.
     + Test 3: The test makes a POST request with missing fields in the payload. It checks if the API returns a status code of 400 (BAD REQUEST) and the read question is not added to the database.
   * Test Explanation: The test ensures that the API behaves correctly for both valid and invalid payloads, and that it properly handles missing fields.
4. AddFavoriteQuestionAPITest:
   * Purpose: This test case validates the add\_favorite\_question API, which allows adding favorite questions for a user.
   * Test Scenario:
     + Test 1: The test makes a valid POST request with a valid payload containing a valid user\_id and question\_id. It checks if the API returns a status code of 201 (CREATED) and the favorite question is added to the database.
     + Test 2: The test makes an invalid POST request with an invalid user\_id (non-existing user). It checks if the API returns a status code of 400 (BAD REQUEST) and the favorite question is not added to the database.
     + Test 3: The test makes a POST request with missing fields in the payload. It checks if the API returns a status code of 400 (BAD REQUEST) and the favorite question is not added to the database.
   * Test Explanation: The test ensures that the API behaves correctly for both valid and invalid payloads, and that it properly handles missing fields.
5. GetFavoriteAndReadQuestionsViewAPITest:
   * Purpose: This test case validates the get\_favorite\_and\_read\_questions API, which retrieves favorite and read questions for a user.
   * Test Scenario: The test sets up test data with favorite and read questions for a user. It then makes a GET request to the API to retrieve the user's favorite and read questions. The test checks if the response contains the correct user information and their respective favorite and read question IDs.
   * Test Explanation: The test verifies that the API returns a status code of 200 (OK) and the correct favorite and read question IDs for the user. It also includes a test to check if the API handles a case where the user is not found, returning a status code of 404 (NOT FOUND) with the appropriate error message.