Software Test Report

Test Report for the Course Controller

- **getAllTutorCourses:**
    1. Test Case 1: Fetching courses for a valid tutor ID
        a. Method: GET
        b. URL: `http://localhost:3000/api/getAllTutorCourses/tutorId`
        c. Expected Response: 200 OK with a list of courses
    2. Test Case 2: Fetching courses for a non-existent tutor ID
        a. Method: GET
        b. URL: `http://localhost:3000/api/getAllTutorCourses/invalidTutorId`
        c. Expected Response: 404 Not Found with an error message
    3. Test Case 3: Fetching courses when no courses are available for the tutor
        a. Method: GET
        b. URL: `http://localhost:3000/api/getAllTutorCourses/noCoursesTutorId`
        c. Expected Response: 404 Not Found with an error message
    4. Test Case 4: Fetching courses with invalid input
        a. Method: GET
        b. URL: `http://localhost:3000/api/getAllTutorCourses/invalidInput`
        c. Expected Response: 400 Bad Request or 404 Not Found with an error message
- **getCourseById:**
    1. Test Case 1: Fetching a course by a valid course ID
        a. Method: GET
        b. URL: `http://localhost:3000/api/getCourseById/courseId`
        c. Expected Response: 200 OK with the course details
    2. Test Case 2: Fetching a course by a non-existent course ID
        a. Method: GET
        b. URL: `http://localhost:3000/api/getCourseById/invalidCourseId`
        c. Expected Response: 404 Not Found with an error message
    3. Test Case 3: Fetching a course with invalid input
        a. Method: GET

    b. URL: `http://localhost:3000/api/getCourseById/invalidInput`
    c. Expected Response: 400 Bad Request or 404 Not Found with an error message

- **getAllStudentCourses:**
  1. Test Case 1: Fetching enrolled courses for a valid student ID
     a. Method: GET
     b. URL: `http://localhost:3000/api/getAllStudentCourses/studentId`
     c. Expected Response: 200 OK with enrolled courses
  2. Test Case 2: Fetching enrolled courses for a non-existent student ID
     a. Method: GET
     b. URL: `http://localhost:3000/api/getAllStudentCourses/invalidStudentId`
     c. Expected Response: 404 Not Found with an error message
  3. Test Case 3: Fetching enrolled courses when no courses are enrolled by the student
     a. Method: GET
     b. URL: `http://localhost:3000/api/getAllStudentCourses/noEnrolledCoursesStudentId`
     c. Expected Response: 404 Not Found with an error message
  4. Test Case 4: Fetching enrolled courses with invalid input
     a. Method: GET
     b. URL: `http://localhost:3000/api/getAllStudentCourses/invalidInput`
     c. Expected Response: 400 Bad Request or 404 Not Found with an error message

- **getAllAvailableCourses:**
  1. Test Case 1: Fetching all available courses when there are available courses
     a. Method: GET
     b. URL: `http://localhost:3000/api/getAllAvailableCourses`
     c. Expected Response: 200 OK with available courses
  2. Test Case 2: Fetching all available courses when no courses are available
     a. Method: GET

b. URL: `http://localhost:3000/api/getAllAvailableCourses/noCourses`
c. Expected Response: 404 Not Found with an error message

3. Test Case 3: Fetching available courses with invalid input
    a. Method: GET
    b. URL: `http://localhost:3000/api/getAllAvailableCourses/invalidInput`
    c. Expected Response: 400 Bad Request or 404 Not Found with an error message.

Test Case for the Flash Card Controller

1. **Testing createFlashCard Endpoint:**

❖ Test Case 1: Create Flash Card Successfully
- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/createFlashCard`
- ➢ Body: (No body required for this test case)
- ➢ Headers:
    - ■ Content-Type: application/json
    - ■ Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 201 Created with success message and created flash card details.

2. **Testing getAllFlashCards Endpoint:**

❖ Test Case 1: Get All Flash Cards Successfully
- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getAllFlashCards`
- ➢ Headers:
    - ■ Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with a list of all flash cards.

3. **Testing getFlashCardByLanguage Endpoint:**

❖ Test Case 1: Get Flash Cards by Language Successfully
- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getFlashCardByLanguage/english`
- ➢ Headers:
    - ■ Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with a list of flash cards filtered by the specified language.

4. **Testing getFlashCardByTutor Endpoint:**

❖ Test Case 1: Get Flash Cards by Tutor Successfully
- ➢ Method: GET

- ➢ URL: `http://localhost:3000/api/getFlashCardByTutor/tutorNameHere`
- ➢ Headers:
  - Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with a list of flash cards filtered by the specified tutor.

5. **Testing getFlashCardById Endpoint:**

❖ Test Case 1: Get Flash Card by ID Successfully

- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getFlashCardById/flashCardIdHere`
- ➢ Headers:
  - Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with the flash card details matching the specified ID.

6. **Testing updateFlashCard Endpoint:**

❖ Test Case 1: Update Flash Card Successfully

- ➢ Method: PUT
- ➢ URL: `http://localhost:3000/api/updateFlashCard/flashCardIdHere`
- ➢ Body: (No body required for this test case)
- ➢ Headers:
  - Content-Type: application/json
  - Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with success message and updated flash card details.

7. **Testing deleteFlashCard Endpoint:**

❖ Test Case 1: Delete Flash Card Successfully

- ➢ Method: DELETE
- ➢ URL: `http://localhost:3000/api/deleteFlashCard/flashCardIdHere`
- ➢ Headers:
  - Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with success message indicating the flash card was deleted.

Test Case for the Student Controller

1. **Testing registerCourse Endpoint:**

❖ Test Case 1: Registering a Course Successfully

- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/registerCourse`
- ➢ Headers:

        - Content-Type: application/json
        - Authorization: Bearer yourAccessTokenHere
    - Expected Response: 200 OK with success message and updated student data.
- Test Case 2: Attempting to Register with Invalid Course ID
    - Method: POST
    - URL: `http://localhost:3000/api/registerCourse`
    - Headers:
        - Content-Type: application/json
        - Authorization: Bearer yourAccessTokenHere
    - Expected Response: 404 Not Found with an error message.
- Test Case 3: Attempting to Register with Invalid Tutor ID
    - Method: POST
    - URL: `http://localhost:3000/api/registerCourse`
    - Headers:
        - Content-Type: application/json
        - Authorization: Bearer yourAccessTokenHere
    - Expected Response: 404 Not Found with an error message.

2. **Testing getMyCourses Endpoint:**

- Test Case 1: Fetching My Courses Successfully
    - Method: GET
    - URL: `http://localhost:3000/api/getMyCourses`
    - Headers:
        - Authorization: Bearer yourAccessTokenHere
    - Expected Response: 200 OK with enrolled courses data.
- Test Case 2: Fetching My Courses without Authorization Token
    - Method: GET
    - URL: `http://localhost:3000/api/getMyCourses`
    - Expected Response: 401 Unauthorised or 403 Forbidden (depends on your authentication setup).

Test Case for the Tutor Controller

1. **Testing isTutor Middleware:**

- Test Case 1: User is a Tutor
    - Method: GET (or any method that triggers the middleware)
    - URL: `http://localhost:3000/api/yourEndpointHere`
    - Headers:
        - Authorization: Bearer yourAccessTokenHere (for a user who is a tutor)
    - Expected Response: Proceeds to the next middleware or endpoint.
- Test Case 2: User is not a Tutor
    - Method: GET (or any method that triggers the middleware)

- ➢ URL: `http://localhost:3000/api/yourEndpointHere`
- ➢ Headers:
  - Authorization: Bearer yourAccessTokenHere (for a user who is not a tutor)
- ➢ Expected Response: 403 Forbidden with an error message indicating the user is not a tutor.

2. **Testing getSchedule Endpoint:**

❖ Test Case 1: Fetching Tutor's Schedule Successfully
- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getSchedule`
- ➢ Headers:
  - Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with the tutor's schedule data.

❖ Test Case 2: Fetching Schedule for a Non-Tutor User
- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getSchedule`
- ➢ Headers:
  - Authorization: Bearer yourAccessTokenHere (for a user who is not a tutor)
- ➢ Expected Response: 404 Not Found with an error message indicating the tutor was not found.

3. **Testing createTutor Endpoint:**

❖ Test Case 1: Creating Tutor Profile Successfully
- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/createTutor`
- ➢ Headers:
  - Content-Type: application/json
  - Authorization: Bearer yourAccessTokenHere
- ➢ Body: (No body required for this test case)
- ➢ Expected Response: 201 Created with success message and created tutor profile details.

❖ Test Case 2: Attempting to Create Tutor Profile with Invalid Data
- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/createTutor`
- ➢ Headers:
  - Content-Type: application/json
  - Authorization: Bearer yourAccessTokenHere
- ➢ Body: (Invalid or missing required fields)
- ➢ Expected Response: 400 Bad Request or 422 Unprocessable Entity with an error message.

4. **Testing getAllTutors Endpoint:**

❖ Test Case 1: Fetching All Tutors Successfully

- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getAllTutors`
- ➢ Headers:
  - ■ Authorization: Bearer yourAccessTokenHere
- ➢ Expected Response: 200 OK with a list of all tutors.

❖ Test Case 2: Fetching Tutors Without Authorization Token

- ➢ Method: GET
- ➢ URL: `http://localhost:3000/api/getAllTutors`
- ➢ Expected Response: 401 Unauthorised or 403 Forbidden (depends on your authentication setup).

Test Case for the Tutor Controller

1. **Testing login Endpoint:**

❖ Test Case 1: Login with Valid Credentials

- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/login`
- ➢ Headers:
  - ■ Content-Type: application/json
- ➢ Body: (No body required for this test case)
- ➢ Expected Response: 200 OK with a success message and user data along with the JWT token in the cookie.

❖ Test Case 2: Login with Invalid Credentials

- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/login`
- ➢ Headers:
  - ■ Content-Type: application/json
- ➢ Body: (Invalid email/password)
- ➢ Expected Response: 401 Unauthorised with an error message indicating invalid credentials.

2. **Testing register Endpoint:**

❖ Test Case 1: Registering a New User Successfully

- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/register`
- ➢ Headers:
  - ■ Content-Type: application/json
- ➢ Body: (User details with a unique email and appropriate role)
- ➢ Expected Response: 200 OK with a success message and user data along with the JWT token in the cookie.

❖ Test Case 2: Attempting to Register as Admin (Not Allowed)

- ➢ Method: POST
- ➢ URL: `http://localhost:3000/api/register`
- ➢ Headers:

        - Content-Type: application/json
    - Body: (User details with role set to ADMIN)
    - Expected Response: 400 Bad Requests with an error message indicating admin registration is not allowed.

3. **Testing logout Endpoint:**

- Test Case: Logging Out
    - Method: GET (or POST if required by your setup)
    - URL: `http://localhost:3000/api/logout`
    - Expected Response: 200 OK with a success message and an expired cookie to clear the JWT token.

4. **Testing protect Middleware:**

- Test Case 1: Accessing Protected Route with Valid Token
    - Method: GET (or any method that requires authentication)
    - URL: `http://localhost:3000/api/protectedEndpoint`
    - Headers:
        - Authorization: Bearer validAccessTokenHere
    - Expected Response: Proceeds to the protected endpoint or middleware.
- Test Case 2: Accessing Protected Route without Token
    - Method: GET (or any method that requires authentication)
    - URL: `http://localhost:3000/api/protectedEndpoint`
    - Expected Response: 401 Unauthorised with an error message indicating no token.

5. **Testing restricTo Middleware:**

- Test Case 1: Restricting Access to Authorised Roles
    - Method: GET (or any method that requires authorization)
    - URL: `http://localhost:3000/api/authorizedEndpoint`
    - Headers:
        - Authorization: Bearer validAccessTokenHere
    - Expected Response: 403 Forbidden with an error message indicating insufficient permissions.
- Test Case 2: Granting Access to Authorised Roles
    - Method: GET (or any method that requires authorization)
    - URL: `http://localhost:3000/api/authorizedEndpoint`
    - Headers:
        - Authorization: Bearer validAccessTokenHere (with a user role that has access)
    - Expected Response: Proceeds to the authorised endpoint or middleware.

Test Case for the Connect Database

1. **Test Case 1: Successful Database Connection**
    a. Description: Connect to a valid MongoDB URI and database name.
    b. Expected Result: Log a message indicating successful connection to the MongoDB database.
2. **Test Case 2: Invalid MongoDB URI**
    a. Description: Provide an invalid MongoDB URI to the `connectDatabase` function.
    b. Expected Result: Log an error message indicating connection failure due to an invalid URI.
3. **Test Case 3: Database Connection Timeout**
    a. Description: Attempt to connect to a MongoDB database that takes longer than the timeout period to respond.
    b. Expected Result: Log an error message indicating connection timeout and exit the process with a non-zero status.
4. **Test Case 4: Database Connection Error**
    a. Description: Connect to a valid MongoDB URI but with incorrect database credentials or permissions.
    b. Expected Result: Log an error message indicating connection failure due to authentication or permission issues and exit the process with a non-zero status.
5. **Test Case 5: Missing MongoDB URI**
    a. Description: Call the `connectDatabase` function without providing a MongoDB URI.
    b. Expected Result: Log an error message indicating that the MongoDB URI is missing and exit the process with a non-zero status.
6. **Test Case 6: Missing Database Name**
    a. Description: Call the `connectDatabase` function without providing a database name.
    b. Expected Result: Log an error message indicating that the database name is missing and exit the process with a non-zero status.

Link: Requirement Traceability Matrics