
# Software Quality Engineering

**DEPARTMENT OF SOFTWARE ENGINEERING (SE)**

**SECTION: BS(SE)-G**

**SUBMITTED TO:**

Mam Saba Kanwal

**SUBMITTED BY:**

Ushna Nadeem
(21i-1225)

Amna Shahid
(21i-1148)

Fatima Qurban
(21i-1195)

**DATE OF SUBMISSION:**

December 11th, 2024

# Backend Files:

### Report Table for Test Cases of Test File: authmiddlewareTest.js

| TestID | Testcase Name | Description | Class/Module | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testNoTokenProvided | Tests behavior when no token is provided | authMiddleware | authMiddleware | Empty req.cookies | 401, "No token, authorization denied" | Same | Pass |
| 2 | testValidToken | Tests behavior when a valid token is provided | authMiddleware | authMiddleware | Valid JWT token | Call next(), req.user set to decoded user | Same | Pass |
| 3 | testInvalidToken | Tests behavior when an invalid token is provided | authMiddleware | authMiddleware | Invalid JWT token | 401, "Token is not valid" | Same | Pass |

---

### Explanation of Coverage Achieved

- **Coverage Achieved:**
    - **Statement Coverage:** All lines in authMiddleware are executed, including:
        - Checking if the token exists.
        - Verifying a valid token.
        - Handling an invalid token with a try-catch block.

    - **Decision Coverage:** All conditional branches are tested:
        - Token existence (true/false).
        - Token validity (valid/invalid).
    - **Exception Handling:** Error handling for an invalid token is verified.
- **Coverage Gaps:**
    - None. All critical scenarios are covered.
- **Improvements:**
    - Additional tests could verify behavior when jwt.verify returns unexpected formats.

## Report Table for Test Cases of Test File: adminRoutesTest.js

| TestID | Testcase Name | Description | Class/Module | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testFetchAllUsers | Tests fetching all users | adminRoutes | GET /users | None | List of users | Same | Pass |
| 2 | testErrorFetchingUsers | Tests error handling for fetching users | adminRoutes | GET /users | None | 500, Error message | Same | Pass |
| 3 | testBlockUser | Tests blocking a user | adminRoutes | PATCH /users/:id/disable | id=1, blocked=true | User unblocked successfully | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | testCannotBlockSelf | Tests that a user cannot block themselves | adminRoutes | PATCH /users/:id/disable | id=1, blocked=true | "Cannot block yourself" | Same | Pass |
| 5 | testInvalidBlockRequest | Tests invalid block requests | adminRoutes | PATCH /users/:id/disable | No blocked in request body | 400, "Bad request" | Same | Pass |
| 6 | testErrorBlockingUser | Tests error handling for user blocking | adminRoutes | PATCH /users/:id/disable | id=1, blocked=true | 500, Error message | Same | Pass |
| 7 | testFetchAllBlogs | Tests fetching all blogs | adminRoutes | GET /blogs | None | List of blogs | Same | Pass |
| 8 | testErrorFetchingBlogs | Tests error handling for fetching blogs | adminRoutes | GET /blogs | None | 500, Error message | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9 | testDisableBlog | Tests disabling a blog | adminRoutes | PATCH /blogs/:id/disable | id=1, block=true | Blog post disabled successfully | Same | Pass |
| 10 | testAlreadyDisabledBlog | Tests handling of already disabled blogs | adminRoutes | PATCH /blogs/:id/disable | id=1, block=true | "Blog post is already disabled" | Same | Pass |
| 11 | testInvalidDisableRequest | Tests invalid disable requests | adminRoutes | PATCH /blogs/:id/disable | No block in request body | 400, "Bad request" | Same | Pass |
| 12 | testErrorDisablingBlog | Tests error handling for blog disabling | adminRoutes | PATCH /blogs/:id/disable | id=1, block=true | 500, Error message | Same | Pass |

---

## Explanation of Coverage Achieved

- **Statement Coverage:** All routes (/users, /blogs) and key code paths are tested, including database operations, error handling, and success cases.
- **Decision Coverage:** All conditions in if blocks (e.g., if (block === null)) are tested, ensuring both true/false paths are covered.
- **Exception Handling:** Error scenarios for database operations are tested comprehensively.

## Report Table for Test Cases of Test File: blogRoutesTest.js

| TestID | Testcase Name | Description | Class | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testCreateBlogPost | Tests creating a blog post | Blog | POST /blogs | Title: "New Blog Post", Content: "Content of the new blog", Author: ID | Blog created successfully | Same | Pass |
| 2 | testGetBlogsExcludingDisabled | Tests retrieving only active blogs | Blog | GET /blogs | N/A | List of active blogs | Same | Pass |
| 3 | testGetAllBlogs | Tests retrieving all blogs, including disabled | Blog | GET /blogs/all | N/A | List of all blogs | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | testGetAuthorBlogs | Tests author-specific blogs with pagination and sorting | Blog | GET /blogs/author/:id | Page: 1, Limit: 1, SortBy: title, SortOrder: asc | Paginated, sorted blogs | Same | Pass |
| 5 | testPatchInvalidAuthor | Tests patching a blog with invalid author | Blog | PATCH /blogs/:id | InvalidAuthor | Unauthorized update attempt | Same | Pass |
| 6 | testDeleteInvalidAuthor | Tests deleting a blog with invalid author | Blog | DELETE /blogs/:id | InvalidAuthor | Unauthorized deletion attempt | Same | Pass |
| 7 | testGetSpecificBlog | Tests fetching a specific blog | Blog | GET /blogs/:id | Valid Blog ID | Blog data | Same | Pass |
| 8 | testRateInvalidRating | Tests rating with an | Blog | PATCH /blogs/:id/rate?rate=6 | Rate: 6 | Rating rejected (Out of range) | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | invalid value | | | | | | |
| 9 | testPostComment | Tests adding a comment to a blog | Blog | POST /blogs/:id/comment | Text: "Another comment", Author: ID | Comment added successfully | Same | Pass |
| 10 | testDeleteComment | Tests deleting a comment from a blog | Blog | DELETE /blogs/:id/comment/:id | Comment ID | Comment deleted successfully | Same | Pass |

---

## Explanation of Coverage Achieved

- **Statement Coverage:** All statements in blogRoutes.js were executed by the test cases.
- **Decision Coverage:** All branches in the code, including true/false conditions, were tested.
- **Loop Coverage:** Loops were tested for zero, one, and multiple iterations (e.g., paginated results).

## Report Table for Test Cases of Test File: searchRoutesTest.js

| TestID | Testcase Name | Description | Class | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | testMissingTextQuery | Tests for missing text query parameter | Blog | find | No text parameter | Empty result set | Same | Pass |
| 2 | testPaginationDefaultSortOrder | Tests pagination with default sort and order | Blog | find | text=test&page=1&limit=2 | 2 results matching query | Same | Pass |
| 3 | testSpecifiedSortOrder | Tests with specified sortBy=content&sortOrder=desc | Blog | find | text=test&sortBy=content&sortOrder=desc | Results sorted by content in descending order | Same | Pass |
| 4 | testInvalidSortOrder | Tests invalid sortBy and sortOrder handling | Blog | find | text=test&sortBy=invalid&sortOrder=invalid | Results sorted by default (title, asce | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | ndin g) | | |
| 5 | testServerError | Tests server error handling | B lo g | find | Simulate database error | 500 statu s code , error mes sage "Dat abas e Erro r" | Sa me | Pas s |
| 6 | testPagination NextPage | Tests pagination next page availability | B lo g | find | text=test&page=1&limit =1 | next obje ct with pag e=2 | Sa me | Pas s |
| 7 | testNoResultsF ound | Tests handling of no matching results | B lo g | find | text=nonexistent | Emp ty resul t set | Sa me | Pas s |

---

## Explanation of Coverage Achieved

- **Statement Coverage:** All executable lines, including sorting, pagination, and exception handling, are covered.
- **Decision Coverage:** All branches, including valid/invalid query parameters and error paths, are tested.

- **Improvement Areas:** Additional tests for edge cases, like invalid limit or page, can further enhance robustness.

## Report Table for Test Cases of Test File: userRoutesTest.js

| TestID | Testcase Name | Description | Class | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testFetchAllUsers | Fetches all users | User | find | GET / | Array of all users | Same | Pass |
| 2 | testFetchSpecificUser | Fetches specific user by ID | User | findById | GET /:id | User object | Same | Pass |
| 3 | testUpdateUserDetails | Updates user details | User | findById | PATCH /:id | Updated user object | Same | Pass |
| 4 | testDeleteUser | Deletes a user by ID | User | findById | DELETE /:id | Success message | Same | Pass |
| 5 | testFollowUser | Adds user to followers and | User | findById | PATCH /:id/follow | Success message | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | following list | | | | | | |
| 6 | testSelfFollow | Handles self-follow attempt | User | findById | PATCH /:id/follow (self) | Error: "You cannot follow yourself" | Same | Pass |

---

## Explanation of Coverage Achieved

- **Statement Coverage:** Every line of code in the router file is covered by the test cases.
- **Decision Coverage:** All possible branches, including valid/invalid inputs and exceptions, are covered.
- **Improvement:** Add edge cases like invalid IDs and malformed requests.

## Report Table for Test Cases of Test File: indexTest.js

| Test ID | Testcase Name | Description | Class | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testCreateUser | Tests user registration | User | register | POST /register with valid user data | User created successfully | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | testMissingFields | Tests user registration with missing fields | User | register | POST /register with missing email | Bad request | Same | Pass |
| 3 | testUserLoginValid | Tests valid user login | User | login | POST /login with correct credentials | Login successful with JWT token | Same | Pass |
| 4 | testUserLoginInvalidEmail | Tests invalid email during login | User | login | POST /login with non-existing email | Cannot find user | Same | Pass |
| 5 | testUserLoginInvalidPass | Tests invalid password during login | User | login | POST /login with incorrect password | Incorrect password | Same | Pass |
| 6 | testUserBlocked | Tests login for blocked user | User | login | POST /login for blocked user | User is blocked | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | testValidTokenAccess | Tests access to protected route with valid token | User | authMiddleware | GET /user with valid JWT token | 200 OK | Same | Pass |
| 8 | testNoTokenAccess | Tests access to protected route without token | User | authMiddleware | GET /user without token | Unauthorized | Same | Pass |
| 9 | testInvalidTokenAccess | Tests access to protected route with invalid token | User | authMiddleware | GET /user with invalid token | Unauthorized | Same | Pass |

---

## Explanation of Coverage Achieved

- **Statement Coverage:** All relevant routes and authentication logic are tested.
- **Decision Coverage:** All decision points, such as valid/invalid inputs, errors, and authorization checks, are covered.
- **Improvement:** Additional edge cases like expired tokens or multiple failed login attempts can be added for further testing.

# Frontend Files:

### Report Table for Test Cases of Test File: AdminTest.jsx

| Test ID | Testcase Name | Description | Class/Module | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testNoTokenProvided | Verifies behavior when no JWT token is provided | Admin.jsx | useEffect | No token in Cookies.get() | Redirects to login | Same | Pass |
| 2 | testFetchBlogs | Checks that blogs are fetched and displayed correctly | Admin.jsx | useEffect | Mock response from axios.get | Blogs displayed with title, author, and ratings | Same | Pass |
| 3 | testHandleSubmit | Tests the blog submission form functionality | Admin.jsx | handleSubmit | Title: "New Blog", Content: "..." | Blog added to list and toast displayed | Same | Pass |

---

## Explanation of Coverage Achieved:

**Statement Coverage:**

- All useEffect hooks executed (fetching blogs, comments, and user).
- All major branches and token checks.

**Decision Coverage:**

- JWT token provided or missing.
- Blog submission form tested for both UI interaction and API calls.

**Coverage Gaps:**

- Error handling during API calls can be explicitly tested.

**Improvements:**

- Add tests for failing API requests to ensure robustness.

## Report Table for Test Cases of Test File: User.jsx

| Test ID | Testcase Name | Description | Class/Module | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|
| 1 | testNoTokenProvided | Tests behavior when no token is provided | Admin_Users | useEffect, fetchUser | No JWT token in cookies | No user data fetched or displayed | Same | Pass |
| 2 | testValidToken | Tests behavior when a valid token is | Admin_Users | useEffect, fetchUser | Valid JWT token | User data fetched and | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | provided | | | | displayed | | |
| 3 | testFetchAdminUsers | Tests if admin users are fetched correctly | Admin_Users | getAdmin_Users | Mocked response with user list | List of users displayed | Same | Pass |
| 4 | testDisableUser | Tests user disabling functionality | Admin_Users | handleDisable | User click event for disabling a user | User blocked status updated and success message | Same | Pass |
| 5 | testDisableUserFailure | Tests error handling when user disable fails | Admin_Users | handleDisable | Error response from backend | Error logged, user state unchanged | Same | Pass |
| 6 | testToastNotification | Verifies that toast notifications are displayed | Admin_Users | handleDisable | Successful user disable action | Toast notification displayed | Same | Pass |

---

## Explanation of Coverage Achieved

**Coverage Achieved:**

- **Statement Coverage**:
    - All lines in the component, including the user fetching (useEffect), admin users fetching (getAdmin_Users), and user disable actions (handleDisable), are executed.
    - Tests ensure that the component renders correctly depending on the presence of the token.
- **Decision Coverage**:
    - Conditional branches are tested, including checking if the token is available and valid, handling the success and failure cases for the user fetching and disabling processes.
- **Exception Handling**:
    - Error handling in the handleDisable function is verified with tests simulating an error response from the backend.

**Coverage Gaps:**

- None identified so far, as all critical paths are covered, including token validation, fetching user data, and handling failures.

**Improvements:**

- Additional tests could verify behavior when fetching user data with no internet connection or when the backend is unresponsive.

## Report Table for Test Cases of Test File: LoginTest.js

| TestID | Testcase Name | Description | Class/Module | Function Tested | Test Inputs | Expected Output | Actual Output | Pass/Fail |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | testRendersForms | Tests if the Login and Signup forms are rendered correctly | Login | Component Rendered | None | Login and SignUp forms rendered | Same | Pass |
| 2 | testLoginEmptyFields | Tests if error is shown when login form fields are empty | Login | handleSubmitLogin | Email: "", Password: "" | Toast error: "Please fill all the fields" | Same | Pass |
| 3 | testSignupEmptyFields | Tests if error is shown when signup form fields are empty | Login | handleSubmitSignup | Username: "", Email: "", Password: "" | Toast error: "Please fill all the fields" | Same | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | testLoginSuccess | Tests if login submits successfully when valid credentials are provided | Login | handleSubmitLogin | Email: "test@test.com", Password: "password" | Toast success: "Logged in successfully" | Same | Pass |
| 5 | testSignupSuccess | Tests if signup submits successfully when valid data is provided | Login | handleSubmitSignup | Username: "testuser", Email: "test@test.com", Password: "password" | Toast success: "Signed up successfully" | Same | Pass |

---

## Explanation of Coverage Achieved

**Coverage Achieved:**

- **Statement Coverage:** All significant code paths in the component are tested:
  - Form rendering
  - Submission handling for both login and signup
  - Error handling for empty fields
  - Success handling for login and signup responses
- **Decision Coverage:**
  - Conditions on field validation for both forms are tested (empty vs. non-empty fields).
  - Valid responses (successful login/signup) are tested.

- **Exception Handling:**
    - Error handling for failed API calls (though not explicitly tested in these cases, this can be added if needed).

**Coverage Gaps:**

- No explicit test for failed login/signup attempts (invalid credentials or server errors). This could be added to complete coverage.

**Improvements:**

- Additional tests could be added to verify behavior in case of network failures or invalid server responses.