IT314: Software Engineering

G17: StaffGrid

JEST for frontend

JEST in frontend code coverage:

```
-------------------|---------|----------|---------|---------|-------------------
File               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------|---------|----------|---------|---------|-------------------
All files          |   89.58 |    71.87 |     100 |   89.58 |
 RegisterEmp.jsx   |   80.76 |     62.5 |     100 |   80.76 | 25-26,55-57
 Salary.jsx        |     100 |       50 |     100 |     100 | 35-58
 UploadProject.jsx |     100 |      100 |     100 |     100 |
-------------------|---------|----------|---------|---------|-------------------
Test Suites: 4 failed, 4 total
Tests:       11 failed, 8 passed, 19 total
Snapshots:   0 total
Time:        5.732 s
Ran all test suites related to changed files.
```

As shown in the snapshot files that do not contain “axios” imports, it can have the best testing strategies. As for files that do contain “axios” imports have low testing strategies because, when testing React components that utilize “axios” to fetch or send data, the core challenge lies in reliably testing asynchronous operations and external dependencies. These can introduce complexities and uncertainties into your test suite.

Approach for not prioritizing “axios” containing files:

1. **Mocking External Dependencies:**

- **Complexity:** Mocking ‘*axios*’ calls can be complex, especially when dealing with complex data structures and error handling.
- **Maintenance Overhead:** As your application evolves, maintaining these mocks can become burdensome.

2. **Test Coverage Limitations:**

- **Asynchronous Nature:** Asynchronous operations, especially those involving network requests, can be difficult to test comprehensively.
- **External Factors:** Factors like network latency, server response times, and API changes can impact test reliability.

3. **Dependency on External Services:**

- **Unreliable Environments:** Relying on external services can make tests flaky and unpredictable.
- **Mock Data Limitations:** Mocked data might not always accurately represent real-world scenarios.

Salary Page:

Test Case ID	Test Description	Testing Steps	Expected Outcome	Remarks
TC-1	Render Salary form	Render the Salary component.	Form should display with "Salary Management Form" and "Employee Salary Details" headings.	Ensures the form renders correctly.
TC-2	Add employees	Click "Add Employee" twice.	Two employee input fields should appear.	Ensures the form dynamically adds employees.
TC-3	Remove an employee	Add two employees and remove one.	Only one employee input field remains.	Ensures employees can be removed from the form.
TC-4	Validate required fields	Add an employee and submit the form without filling any fields.	Error messages appear: "Employee ID is required," "Salary is required," "Bonus is required."	Ensures validation for required fields.
TC-5	Handle optional fields	Add an employee, fill required fields only, and submit the form.	No errors appear, and the form submits successfully.	Ensures optional fields do not block submission.
TC-6	Reject invalid input values	Add an employee, fill in invalid salary (e.g., negative value), and submit the form.	Error message appears: "Salary is required."	Ensures the form validates against invalid input.
TC-7	Submit with multiple employees	Add two employees, fill in their details, and submit the form.	Form logs employee data to console as expected.	Verifies the form handles multiple employees.
TC-8	Submit with no employees	Submit the form without adding employees.	Form logs an empty employee array to console.	Ensures submission is valid with no employees.
TC-9	Handle edge case: remove all employees	Add one employee, fill their details, and then remove them.	All employee input fields disappear, and submission logs an empty employee array.	Ensures edge case is handled gracefully.
TC-10	Handle add, remove, and validation simultaneously	Add employees, leave fields empty, validate errors, correct fields, and resubmit.	Validation errors disappear after correction, and data logs correctly.	Ensures form validation updates dynamically with changes.

Upload Project Page:

Test Case ID	Test Description	Testing Steps	Expected Outcome	Remarks
TC-1	Render project upload form	Render the UploadProject component.	Form should display with fields for title, description, team members, tasks, and a submit button.	Ensures the form renders correctly.
TC-2	Validate required fields	Submit the form without filling in the title or description.	Error messages appear: "Title is required," "Description is required."	Ensures validation for required fields.
TC-3	Add and remove team members	Click "Add Team Member" to add a member and "Remove Member" to remove them.	Team member input fields should appear and disappear accordingly.	Ensures dynamic addition and removal of team members.
TC-4	Add and remove tasks	Click "Add Task" to add a task and "Remove Task" to remove it.	Task input fields (title, assigned ID, description) should appear and disappear accordingly.	Ensures dynamic addition and removal of tasks.
TC-5	Submit valid form data	Fill in valid data for the title and description, and submit the form.	Form logs project data (including title, description, empty team members, and tasks) to console.	Verifies successful form submission with valid inputs.
TC-6	Validate required fields in tasks and team members	Add a team member and a task but do not fill their fields, then submit the form.	Error messages appear: "Member ID is required," "Task title is required," "Assigned ID is required."	Ensures validation for fields in added team members and tasks.
TC-7	Handle start and end dates	Set start and end dates, and verify their values.	Start and end date inputs should hold the entered values.	Ensures proper handling of date inputs.

Register Page:

Test Case ID	Test Description	Testing Steps	Expected Outcome	Remarks
TC-1	Handles valid input for registration	Enter "Ra" for name, "user@example.com" for email, and "manager" for role. Click "Submit".	API calls to email validation and backend registration should succeed.	Validates the happy path for user registration with correct input values.
TC-2	Rejects name exceeding maximum length	Enter a name exceeding 50 characters, e.g., "ThisNameIsVeryLongAndExceedsTheMaxLimit", "user@example.com" for email, and "hr" for role. Click "Submit".	Error message: "Name cannot exceed 50 characters." No API calls.	Ensures the name length validation works as expected.
TC-3	Rejects invalid email format	Enter "Ram Charan" for name, "invalid-email" for email, and "employee" for role. Click "Submit".	Error message: "Please enter a valid email address." No API calls.	Ensures email format validation works before making any API calls.
TC-4	Rejects undeliverable email based on API response	Enter "John Doe" for name, "user@example.com" for email, and "employee" for role. Mock the email validation API to return deliverability as "UNDELIVERABLE". Click "Submit".	Error message: "The email address is invalid or undeliverable." No API calls to the backend.	Ensures that the email validation API result is handled correctly for undeliverable emails.