Full-Stack Simple Web App with Authentication

1. Project Overview

This application is a full-stack authentication system using **React + TypeScript** on the frontend and **Java Spring Boot** on the backend. It supports secure user registration, login, **JWT-based authentication**, and route protection using interceptors and middleware.

2. Identified Potential Bugs or Issues

Frontend:

- Token is stored in localStorage, making it vulnerable to XSS attacks.
- Axios interceptors may cause infinite loops on 401 responses if not handled properly.
- No client-side form validation libraries used, increasing risk of inconsistent form logic.

Backend:

- No explicit password encryption/validation mentioned in code; ensure password hashing (e.g., BCrypt).
- CORS may be incorrectly configured or insecure for production.
- Token expiry logic is not clearly handled; accessToken might remain in storage after expiration.

3. Test Cases

[Login Test]

- Input: Valid credentials
- Expected: Redirect to /home, user info displayed
- Status: ✅ Passed

[Signup Test]

- Input: New user, matching passwords
- Expected: Success message and redirect to login
- Status: ✅ Passed

[Form Validation - Email Format]

- Input: Invalid email (e.g. user@wrong)
- Expected: Show error: "Please enter a valid email address"
- Status: ✅ Passed

[Form Validation - Password Length]

- Input: Password < 6 characters
- Expected: Show error: "Password must be at least 6 characters"
- Status: ✅ Passed

[Form Validation - Confirm Password Mismatch]

- Input: Password = abc123, Confirm Password = abc321
- Expected: Show error: "Passwords do not match"
- Status: ✅ Passed

[Token Expiry]

- Input: Use expired token
- Expected: Redirect to login, logout from app
- Status: ⚠️ Intermittent (manual expiry not handled)

[Unauthorized Access]

- Input: Open /home without login
- Expected: Redirect to /login
- Status: ✅ Passed

[Cross-tab Logout]

- Input: Logout in one tab
- Expected: Another tab also logs out
- Status: ✅ Passed

4. Manual Test Execution Summary

Manual testing covered user registration, login, token storage, logout, and protected routes. All core flows work correctly under expected scenarios. Edge cases like expired tokens and tab sync were tested.

Some limitations observed:

- Expired tokens are not checked automatically.
- UI does not display detailed error messages (e.g., user already exists).

5. SQL Rules

I enforced rules of the database level to ensure triple data validation and integrity, meaning if someone bypasses the front-end and back-end validation, the database would still be secure.

6. Suggestions for Improvement

Frontend:

- Move API base URLs to environment variables.
- Use context or Redux for auth state instead of localStorage polling.
- Integrate `React Hook Form` or `Formik` for form handling.

Backend:

- Add token expiry verification and refresh logic.

- Improve CORS configuration for production.

Testing:

- Write unit tests for services and components (Jest + React Testing Library).
- Add integration tests for backend endpoints using Spring Boot test framework.
- Use Selenium for end-to-end testing

Security:

- Sanitize inputs to prevent XSS/SQL injection.
- Implement HTTPS in production.