Testing Report for Solomonopoly Frontend Project

Overview: This report summarizes the results of running the Jest tests for the Solomonopoly frontend project. The project includes tests for various components, covering functionality such as user registration, login, task management, pack opening, and the user profile page.

Test Suites Included:

1. **App.test.js**: Tests related to the core app rendering.
2. **Home.test.js**: Tests for the homepage components.
3. **Inventory.test.js**: Verifies the inventory management functionality.
4. **LoginForm.test.js**: Validates the login form component.
5. **Map.test.js**: Tests the map-related functionality and rendering.
6. **NotFound.test.js**: Verifies the behaviour of the 404 Not Found page.
7. **PackOpening.test.js**: Ensures the functionality related to pack opening (such as audio, avatars, and animations).
8. **QRScreen.test.js**: Tests QR scanning and camera functionality.
9. **ResetPassword.test.js**: Verifies the password reset feature.
10. **SignUp.test.js**: Tests user registration form validation and functionality.
11. **Splash.test.js**: Verifies the splash screen components.
12. **Store.test.js**: Tests the store functionality for purchasing items with coins.
13. **Taskboard.test.js**: Validates the task management and completion process.
14. **TermsAndConditions.test.js**: Tests the terms and conditions page.

Test Results Summary:

Total Test Suites Run: 14
Test Suites Passed: 5
Test Suites Failed: 9

Key Insights:

1. **Successful Tests**:
   - **App.test.js**: Ran successfully, confirming the app loads the main application components without errors.
   - **Store.test.js**: Validated the basic functionality for the store components such as purchasing packs and viewing items.
   - **Home.test.js**: Tests for the homepage layout and component rendering passed successfully.
2. **Tests Needing Attention**:
   - **LoginForm.test.js**: Encountered issues due to unexpected tokens or misconfigurations in Babel.

- **SignUp.test.js**: Some validations failed because of issues with import.meta syntax which Jest couldn't parse.
- **Taskboard.test.js**: Several test failures related to asynchronous data fetching and state management.
- **QRScreen.test.js**: Camera functionalities and QR scanning failed due to component rendering issues in Jest's testing environment.
- **TermsAndConditions.test.js**: Unable to correctly load TermsAndConditions.jsx due to mock setup errors.
- **ResetPassword.test.js**: Mock functions not configured properly to simulate API requests.

Recommendations:

1. **Jest Configuration**:
   - Make sure import.meta and similar modern syntax is supported in Jest by configuring Babel or adjusting the transform settings to handle ECMAScript Modules (ESM).
   - Mocking external libraries like axios, jwt-decode, or react-router-dom is crucial to ensure isolated testing without relying on API endpoints or navigation.
2. **Mocking Assets**:
   - For images and other assets (e.g., avatars, icons), use Jest's moduleNameMapper configuration to mock them as static files, which will help avoid issues in tests that reference these assets.
3. **Asynchronous Tests**:
   - Ensure that asynchronous operations such as data fetching from APIs or updating state in React components are properly awaited using async/await syntax in test cases.
   - Use the waitFor utility from @testing-library/react to ensure that UI updates or API responses are correctly handled in tests.
4. **Improved Error Handling**:
   - Tests involving external dependencies (e.g., network requests) should have error handling mechanisms in place to capture any failures gracefully.
   - Mock responses for APIs and verify how the components behave when they return error states.

Conclusion:

While several components and functionality tests have passed, a few key tests are failing due to improper Jest configuration, particularly for modern JavaScript syntax and external dependency mocks. Addressing these configuration issues and enhancing mock setups should improve the test suite stability and allow for comprehensive validation of the project's features.