ChatGPT

Preconditions

- **Environment:** An Expo-managed React Native project on **Expo SDK 54** (React Native 0.81) using Expo Router v6. Jest is configured (e.g. with `jest-expo` preset) for unit testing.
- **Dependencies:** As reported, the project has Jest 30.x and `jest-expo` 54.x (e.g. `jest: ^30.2.0`, `jest-expo: ^54.0.13`, plus `@testing-library/react-native`, `react-test-renderer` etc [1]). Confirm all Expo-related packages (including **expo-router**) match SDK 54 versions.
- **Test Location:** Per Expo Router docs, test files **must not** live inside the `/app` (routes) directory. Use a separate `__tests__` folder or similar [2]. Otherwise Expo Router's filesystem conventions can break Jest.
- **Known Issue:** In SDK 54, Expo Router introduced a new internal "winter" runtime. This conflicts with Jest: tests crash with a *ReferenceError* from `expo/src/winter/runtime.native.ts` before any mocks run [3] [4]. In short, the new Expo Router module ("winter") isn't handled by the default Jest setup.

Resolution Attempts (ordered by likely success and simplicity)

1. **Align Jest Dependencies to Expo Recommendations:** Ensure Jest and `jest-expo` versions match Expo's expectations. The Expo upgrade guide indicates using Jest ~29.7.x with `jest-expo` ~54.0.x for SDK 54 [5]. In practice, run:

   ```
   npm install --save-dev jest@29.7.0 jest-expo@~54.0.12
   ```

   This downgrades Jest (and `babel-jest` if needed) to 29.x and pins `jest-expo` to the latest 54.x release (e.g. 54.0.12). Expo Doctor (and the media migration guide) explicitly flagged Jest 30.x and `jest-expo` 53.x as mismatched for SDK 54 [5]. Aligning versions often resolves import/runtime issues. After installing, clear caches (`npm test -- --clearCache`) and rerun Jest.

2. **Modify Jest Transformation Settings:** Adjust the Jest config (`jest.config.js` or `package.json` Jest section) to handle the "winter" files. Two approaches can help:

3. **a. Expand `transformIgnorePatterns`:** Some developers solved similar errors by changing which modules Jest ignores. For example, one solution was to add plain `expo` to `transformIgnorePatterns` (instead of specifically `expo/build/winter`) [6]. In your Jest config, you might ensure that **no Expo modules are inadvertently ignored**. A typical Expo Jest config already includes:

   ```
   "transformIgnorePatterns": [
     "node_modules/(?!(jest-)?react-native|@react-native(-community)?|
   ```

```
expo(nent)?|@expo(nent)?/.*)"
]
```

Verify this includes `expo` and related namespaces. If you had manually added something like `expo/build/winter`, try replacing it with a broader `expo` entry as above [6], or remove it entirely so that Expo modules are transformed.

4. **b. Add a Custom Jest Resolver:** Because "winter" is part of Expo's internal module resolution, another tactic is to use a custom Jest resolver to stub or skip those files. For example, the Nx Expo plugin includes a migration **"add-jest-resolver"** that handles Expo's winter runtime [7]. You could similarly write a `jest.resolver.js` that redirects imports of `expo/src/winter/*` to a dummy module. In `jest.config.js`, specify:

```
module.exports = {
  // ... your existing config ...
  resolver: "<rootDir>/jest.resolver.js"
};
```

And in `jest.resolver.js`, map `expo/src/winter` imports to empty mocks. (The Nx docs note this approach generically: "Add custom Jest resolver to handle Expo winter runtime issues" [7].)

5. **Mock Expo Router/Winter Early:** The error occurs **before** `jest.setup.js` runs, so late mocking won't work. Instead, try mocking at module resolution time. For example, in `jest.config.js` add a `setupFiles` entry (which runs before modules load) that mocks the relevant imports:

```
// jest.config.js
module.exports = {
  // ...
  setupFiles: ['<rootDir>/jest.preload.js']
};
```

And in `jest.preload.js` (a new file), immediately mock Expo router or winter runtime:

```
// jest.preload.js
jest.mock('expo-router', () => ({}));
jest.mock('expo/src/winter/runtime.native', () => {});
```

This ensures when Jest tries to import the winter runtime, it hits a mock. (There's no direct official guide, but mocking at load-time can bypass the ReferenceError.) Test if this lets Jest proceed.

6. **Check for Library Updates or Patches:** Keep an eye on upstream fixes. For example, a known issue in SDK 54 is that `jest-expo`'s preset wasn't updated for the new `expo-file-system` API [8]. The same maintenance lag may apply to Expo Router. Regularly check the expo/expo GitHub issues and jest-expo repo for patches or canary releases. (At the time of writing, expo-file-system required changing imports to `expo-file-system/legacy` [8].) If a new `jest-`

`expo` version is released post-Expo 54, upgrading to that may fix the winter import problem automatically.

7. **Alternative Mocking Strategies (less likely to work):** If the above fail, you could try (a) switching Jest preset from `jest-expo` to `react-native`, or (b) manually editing Babel transforms. However, user reports indicate those approaches were *already attempted and unsuccessful*. Keep them as low priority.

Fallbacks

If unit tests still cannot run under Expo 54, consider these fallbacks:

- **Use Expo SDK 53:** Reverting to **Expo SDK 53** (with `expo-router` v5) avoids the new winter runtime altogether. This is an effective, if heavy-handed, solution. Downgrade with `npx expo install expo@~53.0.0 --fix`, adjust any package versions (see [52]), and re-run your tests. Many developers have postponed using SDK 54 for similar reasons: one user found Expo 54 "the most difficult upgrade" and advised pushing it back until related packages catch up [9]. Downgrading gives Jest back a known-good configuration. (Downside: missing new features and having to re-upgrade later.)

- **Alternative Testing Approaches:** If unit testing with Jest is blocked, shift to other testing strategies for the time being. For instance, use **React Native Testing Library's** integration with Expo Router (the `renderRouter` utility) to write tests that simulate navigation without triggering the winter loader. Or, move critical flows to **E2E tests** (e.g. Detox or Appium). Note that E2E tests require a running simulator/emulator and cover whole-app behavior, not isolated unit logic. This is a fallback if Jest cannot be made to work quickly.

- **Mock/Stubbing:** Another lightweight fallback is to aggressively mock problematic modules. For example, you could stub out the entire `expo-router` or import the legacy versions of modules (`expo-file-system/legacy`) in your code during testing. This might let Jest load without errors but loses actual functionality. It's a last resort and makes tests less meaningful.

- **Manual Testing / TypeScript Safety:** In the extreme case, skip automated tests temporarily. Rely on manual QA, TypeScript typings, and incremental testing. This is risky for long term quality, but sometimes teams proceed with new SDK features and add tests later.

Final Recommendation

Because Expo 54's router "winter" runtime is very new, the simplest path is often to **match Expo's expected versions and retry** (Steps 1–3 above). In practice, start by downgrading Jest to 29.x and `jest-expo` to the latest 54.x (as Expo's doctor suggests) [5], and adjust the Jest config (transformIgnore or custom resolver) to handle `expo/src/winter`. Monitor GitHub for fixes: for example, upcoming `jest-expo` releases may explicitly address this. If business requirements mandate continuous testing now, the safest workaround is to revert to SDK 53 until the ecosystem catches up [10] [9]. Alternative testing (E2E or Expo's integration testing utilities) can fill gaps in the meantime. Each option has trade-offs: downgrading loses new features, custom resolvers add maintenance burden, and skipping Jest reduces code coverage. Choose the approach that best balances

test coverage against development velocity, and stay alert for official updates to Jest/Expo testing support [5] [9].

Sources: Current Expo SDK 54 changelogs and docs [5] [2]; expo/expo GitHub issues (e.g. jest-expo updates) [8]; community reports and migration guides [6] [9] [10].

---

[1] [3] [4] TESTING_BLOCKER.md
file://file_00000000c7dc722f9773e02473e3c4d3

[2] Testing configuration for Expo Router - Expo Documentation
https://docs.expo.dev/router/reference/testing/

[5] Upgrading to React Native 0.81. Upgrading the React Native version is... | by Szvetlin Tanyi | Sep, 2025 | Medium
https://medium.com/@svetlintanyi/upgrading-to-react-native-0-81-4a70149afe80

[6] typescript - Expo Jest fails with Cannot find module 'expo/build/winter' from 'node_modules/jest-expo/src/preset/setup.js' - Stack Overflow
https://stackoverflow.com/questions/77863089/expo-jest-fails-with-cannot-find-module-expo-build-winter-from-node-modules-j

[7] @nx/expo - Migrations | Nx
https://nx.dev/docs/technologies/react/expo/migrations

[8] [jest-expo] New expo-file-system does not work in tests · Issue #39922 · expo/expo · GitHub
https://github.com/expo/expo/issues/39922

[9] Warning after upgrade to Expo54 : r/expo
https://www.reddit.com/r/expo/comments/1nfb24a/warning_after_upgrade_to_expo54/

[10] Trying to set up Expo 54, nothing seems to work : r/expo
https://www.reddit.com/r/expo/comments/1ngqszf/trying_to_set_up_expo_54_nothing_seems_to_work/