The error log shows you're hitting the same core issue as before, but now with `grpc@1.10.1` (an even older version from around 2018) directly under `/node_modules/grpc` — not nested under `google-gax` like in the previous log.

This version of `grpc` (the original native one) is **long deprecated**, has no prebuilt binaries for Node 20 (ABI v115), and fails to compile from source on modern macOS/Xcode/clang due to strict warnings treated as errors (e.g., `-Wunused-but-set-variable` in `channel_stack.cc`).

Old `grpc@1.x` simply doesn't support Node.js 14+, let alone 20.19. The replacement is `@grpc/grpc-js` (pure JS, no compilation needed, actively maintained).

## Quick Diagnosis Steps

Run these in your project root to find the culprit:

```
Bash

npm ls grpc
npm ls google-gax   # often the indirect cause
npm why grpc        # shows the dependency tree path
```

Common culprits in React projects (especially older ones or those with backend integrations):

- Old versions of `@google-cloud/*` packages (e.g., Firestore, Pub/Sub, BigQuery via `google-gax`)
- Very old `firebase-admin` (Node server SDK — but unlikely in pure React unless you're doing SSR/Next.js with admin features)
- Some niche gRPC client libs or outdated Google API clients

Paste the output of `npm ls grpc` if you can — it'll pinpoint exactly what's pulling it in.

## Fix Options (Start with #1 or #2)

1. **Override the bad grpc with modern @grpc/grpc-js** (best first try — works for many old google-gax usages)

   In your `package.json`, add a resolutions section (npm/yarn/pnpm all support this):

   ```JSON
   "resolutions": {
     "grpc": "@grpc/grpc-js"
   }
   ```

   - If `google-gax` is present and stubborn, also add:

     ```JSON
     "google-gax/grpc": "@grpc/grpc-js"
     ```

   Then clean and reinstall:

   ```Bash
   rm -rf node_modules package-lock.json   # (or yarn.lock / pnpm-lock.yaml
   npm cache clean --force
   npm install
   ```

   This forces everything to use the JS implementation instead of the ancient native one.

2. **Update the offending package(s)**

   Once you know what's depending on `grpc@1.10.1` (from `npm why grpc` or `npm ls`), update it:

   - If it's a `@google-cloud/something` → `npm install @google-cloud/<package>@latest`
   - If it's old `firebase` / `firebase-admin` → update to latest (they switched to `@grpc/grpc-js` years ago)
   - Or run a dependency update tool:

   ```
   Bash

   npx npm-check-updates -u
   npm install
   ```

   Then fix any breaking changes (often minimal for Google libs).

3. **Temporary workaround: Downgrade Node (only to unblock you quickly)**

   Since you're using nvm:

   ```
   Bash

   nvm install 16   # or 14 if 16 still fails
   nvm use 16
   rm -rf node_modules package-lock.json
   npm install
   ```

   This often lets old grpc@1.10 compile, but **don't stay here** — Node 16 is EOL, and you'll hit security/support issues soon. Use this only to get running while you fix deps.

4. **If nothing depends on it meaningfully → Remove it**

If `npm ls grpc` shows it's not actually used (transitive dep you don't need), you can add to `package.json`:

```JSON
"dependencies": {
  // your stuff
},
"overrides": {
  "grpc": false   // or use resolutions to null it if possible
}
```

But overrides/resolutions to `@grpc/grpc-js` is safer.

## Additional Notes

- The `core-js@2.5.5` deprecation is unrelated and harmless — it's from an old Babel/polyfill setup. You can ignore or update `core-js` later via `npm update core-js`.
- Your macOS (Darwin 24.6.0 → likely macOS 15 Sequoia) + Xcode 16+ clang is stricter with C++ warnings, which is why even small unused variables fail under `-Werror`.
- If this is a Create React App or Vite project with no server/gRPC needs → something like an old tutorial/boilerplate probably added Google Cloud deps accidentally.

Try the resolutions trick first — it's fixed this exact grpc@1.x nightmare for tons of people on Node 18/20. If it still fails, share the `npm ls grpc` output or what package is pulling it in, and we can get more specific.