🔍 Build and Runtime Issues Investigation Report

Date: November 13, 2025
Repository: UPRISE_NEXT
Commit Investigated: 3a33491 (fix(api): Override noEmit from base tsconfig to enable build output)
Agent: DeepAgent

📋 Executive Summary

The investigation confirmed **three critical issues** with the NestJS API build configuration:

1. ✅ **CONFIRMED:** Build output directory nesting issue (dist/apps/api/src/ instead of dist/)
2. ✅ **CONFIRMED:** Module resolution errors with @uprise/types imports at runtime
3. ✅ **CONFIRMED:** NestJS build configuration needs optimization

The API **builds successfully** but **cannot start** due to incorrect file paths and module resolution failures.

🔬 Investigation Findings

Issue #1: Build Output Directory Nesting

Expected Behavior:

```
apps/api/dist/
├── main.js
├── app.module.js
├── communities/
│   ├── communities.controller.js
│   └── communities.service.js
└── ... (other modules)
```

Actual Behavior:

```
apps/api/dist/
├── apps/
│   └── api/
│       └── src/
│           ├── main.js
│           ├── app.module.js
│           └── ... (nested structure)
├── packages/
│   └── types/
│       └── src/
│           └── ... (compiled types)
└── tsconfig.tsbuildinfo
```

Root Cause:
The TypeScript compiler is preserving the entire monorepo directory structure when compiling. This happens because:

1. The `baseUrl` in `tsconfig.json` is set to `"./"` (current directory)
2. TypeScript path mappings point to `../../packages/types/src`
3. The compiler maintains the relative path structure from the baseUrl to the source files
4. NestJS's `nest build` command uses TypeScript's compiler which respects these settings

Impact:
- The start script `"node dist/main"` fails because it expects `dist/main.js` but finds `dist/apps/api/src/main.js`
- Deployment configurations expecting a flat dist structure will fail
- Increased bundle size due to unnecessary directory nesting

Issue #2: Module Resolution Errors

Error Message:

```
Error [ERR_MODULE_NOT_FOUND]: Cannot find module '/home/ubuntu/UPRISE_NEXT/packages/types/src/user'
imported from /home/ubuntu/UPRISE_NEXT/packages/types/src/index.ts
```

Root Cause:
The TypeScript path mapping configuration is not being translated to runtime module resolution:

1. **Compile Time:** TypeScript successfully resolves `@uprise/types` to `../../packages/types/src` during compilation
2. **Runtime:** Node.js doesn't understand TypeScript path mappings
3. **Import Resolution:** When the compiled code tries to import from the compiled `@uprise/types` package, it references the original `.ts` source files, not the compiled `.js` files
4. **Missing .js Extension:** The compiled code imports `./user` instead of `./user.js`

Example from compiled code:

```
// In dist/packages/types/src/index.js
__exportStar(require("./user"), exports);  // ❌ Should be "./user.js"
```

Why This Happens:
- TypeScript doesn't rewrite relative imports to add `.js` extensions when targeting CommonJS
- The packages/types compiled output is included in dist/, but the imports still reference .ts files
- pnpm workspace protocol `"@uprise/types": "workspace:*"` is not resolved at runtime

Issue #3: NestJS Build Configuration

Current Configuration Analysis:

apps/api/nest-cli.json:

```
{
  "$schema": "https://json.schemastore.org/nest-cli",
  "collection": "@nestjs/schematics",
  "sourceRoot": "src",
  "compilerOptions": {
    "deleteOutDir": true
  }
}
```

apps/api/tsconfig.json:

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "module": "commonjs",
    "outDir": "./dist",
    "baseUrl": "./",
    "noEmit": false,  // ✅ Correctly overridden
    "paths": {
      "@uprise/types": ["../../packages/types/src"]  // ⚠️ Issue
    }
  }
}
```

Problems Identified:

1. No `rootDir` specified, causing TypeScript to infer it from all input files
2. Path mappings point to source files (.ts) instead of compiled output or package main
3. `baseUrl` set to "./" causes relative path preservation
4. Missing `assets` configuration in nest-cli.json for Prisma schema files
5. No webpack or build hooks to transform path aliases

🛠️ Proposed Solutions

Solution #1: Fix Build Output Directory Structure

Option A: Configure rootDir (Recommended)

Update `apps/api/tsconfig.json`:

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "module": "commonjs",
    "moduleResolution": "node",
    "target": "ES2021",
    "outDir": "./dist",
    "rootDir": "./src",  // ✅ Add this
    "baseUrl": "./",
    "noEmit": false,
    "paths": {
      "@uprise/types": ["../../packages/types/src"]
    }
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist"]
}
```

What this fixes:
- Forces TypeScript to treat `src/` as the root
- Outputs files relative to src/, so `src/main.ts` becomes `dist/main.js`
- Eliminates the nested directory structure

Update package.json start script:

```
{
  "scripts": {
    "start": "node dist/main.js"
  }
}
```

Solution #2: Fix Module Resolution Errors

Option A: Use Compiled Output for Types Package (Recommended)

1. **Build the types package first:**

Create `packages/types/tsconfig.json`:

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "outDir": "./dist",
    "rootDir": "./src",
    "declaration": true,
    "noEmit": false
  },
  "include": ["src/**/*"]
}
```

Update `packages/types/package.json`:

```
{
  "name": "@uprise/types",
  "version": "1.0.0",
  "private": true,
  "main": "./dist/index.js",      // ✅ Point to compiled output
  "types": "./dist/index.d.ts",   // ✅ Point to compiled types
  "scripts": {
    "build": "tsc",
    "lint": "eslint . --ext .ts",
    "typecheck": "tsc --noEmit"
  }
}
```

1. **Update API tsconfig to reference compiled output:**

Update `apps/api/tsconfig.json`:

```
{
  "compilerOptions": {
    "paths": {
      "@uprise/types": ["../../packages/types/dist"]  // ✅ Point to dist
    }
  }
}
```

1. **Update turbo.json to build types first:**

```
{
  "tasks": {
    "build": {
      "dependsOn": ["^build"],  // Already correct - ensures packages build first
      "outputs": [".next/**", "!.next/cache/**", "dist/**", "build/**"]
    }
  }
}
```

Option B: Use tsconfig-paths at Runtime (Alternative)

Install and configure tsconfig-paths:

```
cd apps/api
pnpm add tsconfig-paths
```

Update start script in `apps/api/package.json`:

```
{
  "scripts": {
    "start": "node -r tsconfig-paths/register dist/main.js"
  }
}
```

Create `apps/api/tsconfig.paths.json`:

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@uprise/types": ["../../packages/types/src"]
    }
  }
}
```

Option C: Remove Path Aliases and Use Relative Imports (Last Resort)

Replace all `@uprise/types` imports with relative imports:

```
// Before
import { User } from '@uprise/types';

// After
import { User } from '../../packages/types/src';
```

Solution #3: Optimize NestJS Build Configuration

Complete nest-cli.json Configuration:

```
{
  "$schema": "https://json.schemastore.org/nest-cli",
  "collection": "@nestjs/schematics",
  "sourceRoot": "src",
  "compilerOptions": {
    "deleteOutDir": true,
    "assets": [
      {
        "include": "../prisma/**/*",
        "outDir": "./dist",
        "watchAssets": true
      }
    ],
    "watchAssets": true
  }
}
```

Complete apps/api/tsconfig.json:

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "module": "commonjs",
    "moduleResolution": "node",
    "declaration": true,
    "removeComments": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "allowSyntheticDefaultImports": true,
    "target": "ES2021",
    "sourceMap": true,
    "outDir": "./dist",
    "rootDir": "./src",
    "baseUrl": "./",
    "incremental": true,
    "skipLibCheck": true,
    "strictNullChecks": true,
    "noImplicitAny": true,
    "strictBindCallApply": true,
    "forceConsistentCasingInFileNames": true,
    "noFallthroughCasesInSwitch": true,
    "noEmit": false,
    "paths": {
      "@uprise/types": ["../../packages/types/dist"]
    }
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist", "test", "**/*spec.ts"]
}
```

🎯 Recommended Implementation Plan

Phase 1: Fix Critical Build Issues (Priority: HIGH)

1. **Add rootDir to apps/api/tsconfig.json**
   - Fixes the nested directory structure
   - Ensures dist/main.js is at the correct location
   - Estimated time: 5 minutes

2. **Build types package separately**
   - Create packages/types/tsconfig.json
   - Update packages/types/package.json to point to dist/
   - Add build script to packages/types
   - Estimated time: 10 minutes

3. **Update API path mappings**
   - Change @uprise/types path to point to ../../packages/types/dist
   - Estimated time: 2 minutes

4. **Update package.json scripts**
   - Ensure start script points to dist/main.js
   - Add prebuild script to build dependencies
   - Estimated time: 5 minutes

Phase 2: Test and Validate (Priority: HIGH)

1. **Clean build test**
   ```
   bash
   cd /home/ubuntu/UPRISE_NEXT
   rm -rf apps/api/dist packages/types/dist
   pnpm install
   cd packages/types && pnpm run build
   cd ../../apps/api && pnpm run build
   pnpm run start
   ```

2. **Verify endpoints**
   - GET /api/health
   - GET /api/health/postgis
   - GET /api/communities (if database is set up)

3. **Run tests**
   ```
   bash
   cd apps/api
   pnpm run test
   ```

Phase 3: Optimize Configuration (Priority: MEDIUM)

1. **Update nest-cli.json** with asset management
2. **Add development convenience scripts**
3. **Update documentation** in README.md

📊 Testing Results

Build Test

- **Status:** ✅ SUCCESS
- **Command:** `pnpm run build`
- **Output:** Compiled successfully to `dist/apps/api/src/`
- **Issues:** Output directory structure is nested (see Issue #1)

Start Test

- **Status:** ❌ FAILURE
- **Command:** `pnpm run start`
- **Error:**
  ```
  Error: Cannot find module '/home/ubuntu/UPRISE_NEXT/apps/api/dist/main'
  ```
- **Reason:** Expected file at dist/main.js but actual location is dist/apps/api/src/main.js

Start Test (Direct Path)

- **Status:** ❌ FAILURE
- **Command:** `node dist/apps/api/src/main.js`
- **Error:**
  ```
  Error [ERR_MODULE_NOT_FOUND]: Cannot find module '/home/ubuntu/UPRISE_NEXT/packages/types/src/user'
  ```
- **Reason:** Module resolution fails for @uprise/types package (see Issue #2)

Endpoint Tests

- **Status:** ⏭️ SKIPPED
- **Reason:** API server cannot start due to module resolution errors

---

📁 Current Configuration Files

apps/api/tsconfig.json (Current)

```
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "module": "commonjs",
    "moduleResolution": "node",
    "declaration": true,
    "removeComments": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "allowSyntheticDefaultImports": true,
    "target": "ES2021",
    "sourceMap": true,
    "outDir": "./dist",
    "baseUrl": "./",
    "incremental": true,
    "skipLibCheck": true,
    "strictNullChecks": true,
    "noImplicitAny": true,
    "strictBindCallApply": true,
    "forceConsistentCasingInFileNames": true,
    "noFallthroughCasesInSwitch": true,
    "noEmit": false,
    "paths": {
      "@uprise/types": ["../../packages/types/src"]
    }
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist"]
}
```

apps/api/nest-cli.json (Current)

```
{
  "$schema": "https://json.schemastore.org/nest-cli",
  "collection": "@nestjs/schematics",
  "sourceRoot": "src",
  "compilerOptions": {
    "deleteOutDir": true
  }
}
```

tsconfig.base.json (Root - Current)

```
{
  "$schema": "https://json.schemastore.org/tsconfig",
  "display": "Base",
  "compilerOptions": {
    "composite": false,
    "declaration": true,
    "declarationMap": true,
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "inlineSources": false,
    "isolatedModules": true,
    "moduleResolution": "bundler",
    "noUnusedLocals": false,
    "noUnusedParameters": false,
    "preserveWatchOutput": true,
    "skipLibCheck": true,
    "strict": true,
    "strictNullChecks": true,
    "resolveJsonModule": true,
    "allowJs": true,
    "noEmit": true,
    "incremental": true,
    "target": "ES2022",
    "module": "ESNext",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
  },
  "exclude": ["node_modules"]
}
```

🚀 Quick Fix Commands

To implement all recommended fixes, run:

```
cd /home/ubuntu/UPRISE_NEXT

# 1. Create types package tsconfig
cat > packages/types/tsconfig.json << 'TYPES_EOF'
{
  "extends": "../../tsconfig.base.json",
  "compilerOptions": {
    "outDir": "./dist",
    "rootDir": "./src",
    "declaration": true,
    "declarationMap": true,
    "noEmit": false,
    "module": "commonjs",
    "target": "ES2021"
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "dist"]
}
TYPES_EOF

# 2. Update types package.json main and types fields
cd packages/types
npm pkg set main="./dist/index.js"
npm pkg set types="./dist/index.d.ts"
npm pkg set scripts.build="tsc"
cd ../..

# 3. Add rootDir to API tsconfig and update paths
cd apps/api
# Manual edit required - see Solution #1 and #2 above

# 4. Build types package
cd ../../packages/types
pnpm run build

# 5. Build and start API
cd ../../apps/api
rm -rf dist
pnpm run build
pnpm run start
```

📝 Additional Recommendations

1. Add Build Scripts to Root package.json

```
{
  "scripts": {
    "build": "turbo run build",
    "build:types": "pnpm --filter @uprise/types build",
    "build:api": "pnpm --filter api build",
    "build:clean": "rm -rf apps/*/dist packages/*/dist && pnpm build"
  }
}
```

2. Update .gitignore

Ensure dist directories are ignored:

```
**/dist/
**/.turbo/
**/node_modules/
**/*.tsbuildinfo
```

3. Add Pre-build Hook

In `apps/api/package.json`:

```
{
  "scripts": {
    "prebuild": "cd ../../packages/types && pnpm run build",
    "build": "nest build"
  }
}
```

4. Consider Production Build Optimization

For production deployments:
- Use `NODE_ENV=production`
- Enable tree-shaking with proper module settings
- Consider using webpack for better module resolution
- Add startup script that checks for required compiled dependencies

---

🔗 Related Files and References

Files Examined

- `/home/ubuntu/UPRISE_NEXT/apps/api/tsconfig.json`
- `/home/ubuntu/UPRISE_NEXT/apps/api/nest-cli.json`
- `/home/ubuntu/UPRISE_NEXT/apps/api/package.json`
- `/home/ubuntu/UPRISE_NEXT/tsconfig.base.json`
- `/home/ubuntu/UPRISE_NEXT/turbo.json`
- `/home/ubuntu/UPRISE_NEXT/packages/types/package.json`

Build Output Structure

```
apps/api/dist/
├── apps/
│   └── api/
│       └── src/
│           ├── main.js
│           ├── app.module.js
│           ├── auth/
│           ├── communities/
│           ├── events/
│           ├── health/
│           ├── prisma/
│           ├── tracks/
│           └── users/
├── packages/
│   └── types/
│       └── src/
│           ├── index.js
│           ├── user.js
│           ├── community.js
│           ├── track.js
│           ├── event.js
│           ├── auth.js
│           └── api.js
└── tsconfig.tsbuildinfo
```

✅ Verification Checklist

After implementing fixes, verify:

- [ ] API builds successfully without errors
- [ ] Build output is at `dist/main.js` (not `dist/apps/api/src/main.js`)
- [ ] API starts successfully with `pnpm run start`
- [ ] No module resolution errors at runtime
- [ ] @uprise/types imports work correctly
- [ ] Health endpoints respond successfully
- [ ] PostGIS endpoints work (GET /communities, POST /communities)
- [ ] Tests pass successfully
- [ ] Dev mode works (`pnpm run dev`)

🏁 Conclusion

All three issues have been identified with clear root causes:

1. **Build Output Nesting:** Missing `rootDir` configuration causes TypeScript to preserve full directory structure
2. **Module Resolution:** Path mappings point to .ts source files instead of compiled .js output
3. **NestJS Configuration:** Missing optimization settings and asset management

The recommended fixes are straightforward and can be implemented in ~30 minutes. The highest priority is fixing the rootDir and types package build configuration to enable the API to start successfully.

Next Steps:
1. Implement the recommended fixes in Phase 1
2. Test thoroughly in Phase 2
3. Update documentation
4. Commit changes with proper git tags

Generated by: DeepAgent
Report Date: November 13, 2025
Status: Ready for Implementation