Heroku Buildpack Deployment Fix

Problem Summary

The application was failing to deploy on Heroku/Easypanel with the following error:

```
Error: Cannot find module '/workspace/.next/standalone/server.js'
```

Despite successful builds, the application would crash at runtime because the `.next/standalone/` directory was not being included in the final container image.

Root Cause

Next.js Standalone Mode is incompatible with Heroku buildpacks.

Why Standalone Mode Failed

1. **Build Output Location**: When using `output: 'standalone'` in `next.config.js`, Next.js creates a minimal production build in `.next/standalone/` with only necessary files.

2. **Heroku Buildpack Behavior**: The official Heroku Node.js buildpack (and Easypanel which uses Heroku buildpacks) doesn't recognize or copy the `.next/standalone/` directory structure to the runtime container.

3. **Missing Files**: While the build process succeeds, the buildpack copies the standard `.next/` directory but excludes the `standalone/` subdirectory, resulting in missing `server.js` file at runtime.

4. **Container Architecture**: Heroku buildpacks are designed to work with standard Next.js deployments, not the experimental standalone mode which was introduced for Docker-based deployments.

Solution

Use Standard Next.js Mode instead of standalone mode with Heroku buildpacks.

Changes Made

1. `next.config.js`

Before:

```
const nextConfig = {
  output: 'standalone',  // ❌ Not compatible with Heroku buildpacks
  images: {
    // ...
  }
};
```

After:

```
const nextConfig = {
  // Removed 'output: standalone' - not compatible with Heroku buildpacks
  images: {
    // ...
  }
};
```

2. `package.json`

Before:

```
{
  "scripts": {
    "start": "node .next/standalone/server.js"  // ❌ Wrong for standard mode
  }
}
```

After:

```
{
  "scripts": {
    "start": "next start"  // ✅ Standard Next.js start command
  }
}
```

3. `Procfile`

Before:

```
web: node .next/standalone/server.js  # ❌ Wrong path
```

After:

```
web: npm start  # ✅ Use npm start which runs "next start"
```

When to Use Standalone Mode

Standalone mode should ONLY be used when:

1. **Docker Deployments**: You're building a custom Docker image where you control the entire containerization process
2. **Custom Deployment Pipelines**: You have a custom CI/CD that explicitly copies the standalone output
3. **Self-Hosted Environments**: You're deploying to environments where you manage the entire file structure

When to Use Standard Mode

Use standard Next.js mode when deploying to:

1. **Heroku** (official Heroku platform)

2. **Easypanel** (uses Heroku buildpacks)
3. **Platform.sh**
4. **Railway** (if using buildpack mode)
5. **Any platform using Heroku buildpacks**

Deployment Instructions for Easypanel

Prerequisites

- Node.js version specified in `package.json` or `.node-version` file
- PostgreSQL database configured
- Environment variables set (DATABASE_URL, NEXTAUTH_SECRET, etc.)

Deployment Steps

1. **Push Code to GitHub**
   ```
   bash
      git add .
      git commit -m "Fix: Remove standalone mode for Heroku buildpack compatibility"
      git push origin master
   ```

2. **In Easypanel**
   - Connect your GitHub repository
   - Select branch: `master`
   - Build method: Heroku Buildpack (automatic detection)
   - The Procfile will be automatically detected

3. **Environment Variables**
   Set the following in Easypanel:
   ```
   DATABASE_URL=your_postgres_connection_string
      NEXTAUTH_SECRET=your_secret_key
      NEXTAUTH_URL=your_production_url
      AWS_ACCESS_KEY_ID=your_aws_key
      AWS_SECRET_ACCESS_KEY=your_aws_secret
      AWS_REGION=your_aws_region
      AWS_S3_BUCKET_NAME=your_bucket_name
      NODE_ENV=production
   ```

4. **Deploy**
   - Trigger deployment
   - Monitor build logs to ensure successful build
   - Application should start successfully with `next start`

Expected Build Output

With standard mode, you should see:

```
✓ Creating an optimized production build
✓ Compiled successfully
✓ Generating static pages
✓ Finalizing page optimization
```

NOT:

```
Automatically copying files for standalone output
```

Verification

After deployment, verify:

1. **Build Logs**: Should show “Compiled successfully”
2. **Startup Logs**: Should show Next.js server starting on the configured port
3. **No Errors**: Should not see “Cannot find module” errors
4. **Application Access**: Should be able to access the application URL

Performance Considerations

Q: Is standard mode slower than standalone mode?

A: Not significantly. While standalone mode produces a smaller output (only necessary files), standard mode includes all files but:
- Still optimized for production
- Uses Next.js built-in optimizations
- Comparable performance in most cases
- Better compatibility with deployment platforms

Additional Resources

- Next.js Output File Tracing (https://nextjs.org/docs/advanced-features/output-file-tracing)
- Heroku Node.js Buildpack (https://github.com/heroku/heroku-buildpack-nodejs)
- Next.js Deployment Documentation (https://nextjs.org/docs/deployment)

Troubleshooting

If build still fails:

1. **Clear buildpack cache**
   `bash`
   `   # In Easypanel, trigger a rebuild with cache cleared`

2. **Verify Node version**
   - Check that Node.js version is compatible (14.6.0 or higher)
   - Add `.node-version` file if needed

3. **Check dependencies**
   - Ensure all dependencies are in `dependencies`, not `devDependencies`
   - Run `npm install` locally to verify

4. **Environment variables**
   - Verify all required environment variables are set
   - Check DATABASE_URL connection string format

Last Updated: October 27, 2025
Status: ✅ Fixed and Verified