Vercel Deployment Fixes - Documentation

Overview

This document outlines the fixes applied to resolve Vercel deployment issues for the Biospark Health AI application.

Issues Addressed

1. Primary Issue: Prisma Client Generation

Problem: Vercel caches dependencies between builds, causing Prisma Client to become outdated and fail initialization.

Error Message:

```
Prisma has detected that this project was built on Vercel, which caches dependencies.
This leads to an outdated Prisma Client because Prisma's auto-generation isn't
triggered.
```

Solution Applied:
- Added `"postinstall": "prisma generate"` to package.json scripts
- Modified build script to include `"build": "prisma generate && next build"`
- This ensures Prisma Client is regenerated on every build, regardless of Vercel's caching

2. Edge Runtime Crypto Module Issue

Problem: Node.js 'crypto' module not supported in Edge Runtime environments.

Error Message:

```
The edge runtime does not support Node.js 'crypto' module.
```

Solution Applied:
- Implemented conditional crypto import in `lib/crypto.ts`
- Added Web Crypto API fallback for Edge Runtime compatibility
- Created both synchronous (Node.js) and asynchronous (Edge Runtime) versions of encryption methods
- Added proper error handling with informative messages for unsupported operations

Key Changes in lib/crypto.ts:

```
// Conditional crypto import
let crypto: any;
let isNodeEnvironment = false;

try {
  crypto = require('crypto');
  isNodeEnvironment = true;
} catch (error) {
  crypto = globalThis.crypto;
  isNodeEnvironment = false;
}
```

3. Security Vulnerabilities

Issues Found:
- 1 low severity vulnerability in Next.js (< 14.2.30)
- 2 critical vulnerabilities in @getzep/zep-js and form-data

Solutions Applied:
- Updated Next.js from 14.2.28 to 14.2.30
- Downgraded @getzep/zep-js from ^2.0.2 to ^0.10.0 (stable version)
- Ran `npm audit fix` to resolve form-data vulnerability
- All vulnerabilities now resolved (0 vulnerabilities remaining)

4. Vercel Configuration Optimization

Changes Made:
- Removed `--force` flags from install and build commands in vercel.json
- Simplified build process to rely on package.json scripts
- Updated configuration to use standard npm commands

Files Modified

1. package.json

```
{
  "scripts": {
    "build": "prisma generate && next build",
    "postinstall": "prisma generate",
    // ... other scripts
  },
  "dependencies": {
    "next": "14.2.30",
    "@getzep/zep-js": "^0.10.0",
    // ... other dependencies
  }
}
```

2. lib/crypto.ts

- Added conditional crypto import for Edge Runtime compatibility
- Implemented Web Crypto API fallback methods
- Added async versions of all encryption/decryption methods
- Maintained backward compatibility with existing synchronous methods

3. vercel.json

```
{
  "buildCommand": "npm run build",
  "installCommand": "npm ci",
  // ... other configuration
}
```

Verification Steps

Local Testing

1. ✅ `npm ci` - Dependencies install successfully with Prisma generation
2. ✅ `npm run build` - Build completes successfully
3. ✅ `npm audit` - No security vulnerabilities found
4. ✅ Crypto module works in both Node.js and Edge Runtime contexts

Expected Vercel Deployment Behavior

1. **Install Phase**: `npm ci` will install dependencies and trigger `postinstall` script
2. **Build Phase**: `npm run build` will run `prisma generate && next build`
3. **Result**: Fresh Prisma Client generated on every deployment, preventing cache-related issues

Best Practices Implemented

1. Prisma + Vercel

- Always regenerate Prisma Client during build process
- Use `postinstall` script as backup for dependency caching scenarios
- Ensure `prisma` is in dependencies (not just devDependencies)

2. Edge Runtime Compatibility

- Conditional imports for environment-specific APIs
- Graceful fallbacks for unsupported operations
- Clear error messages for debugging

3. Security

- Regular dependency updates
- Vulnerability scanning and resolution
- Use of stable package versions

Environment Variables Required

For successful deployment, ensure these environment variables are set in Vercel:

```
# Database
DATABASE_URL="postgresql://..."
DIRECT_URL="postgresql://..."

# Authentication
NEXTAUTH_SECRET="..."
NEXTAUTH_URL="https://your-domain.vercel.app"

# Encryption
PHI_ENCRYPTION_KEY="..."
AUDIT_SALT="..."

# External Services
ZEP_API_KEY="..."
ZEP_API_URL="..."
OPENAI_API_KEY="..."
NEXT_PUBLIC_SUPABASE_URL="..."
NEXT_PUBLIC_SUPABASE_ANON_KEY="..."
SUPABASE_SERVICE_ROLE_KEY="..."
REDIS_URL="..."
```

Deployment Checklist

Before deploying to Vercel:

- [ ] All environment variables configured in Vercel dashboard
- [ ] Database accessible from Vercel's deployment regions
- [ ] Redis instance configured and accessible
- [ ] External API keys valid and have proper permissions
- [ ] Domain configured correctly for NEXTAUTH_URL

Troubleshooting

If Prisma Issues Persist

1. Check that `DATABASE_URL` is correctly set in Vercel environment variables
2. Verify database is accessible from Vercel's deployment regions
3. Check Vercel build logs for specific Prisma error messages

If Edge Runtime Issues Occur

1. Verify that API routes using crypto are not forced to Edge Runtime
2. Check that async crypto methods are used in Edge contexts
3. Review middleware for Node.js-specific imports

If Build Fails

1. Check Vercel build logs for specific error messages
2. Verify all required environment variables are set
3. Test build locally with production environment variables

Next Steps

1. Deploy to Vercel and monitor build logs

2. Test all functionality in production environment
3. Monitor for any runtime errors related to crypto operations
4. Set up proper monitoring and alerting for production issues

---

Date: July 22, 2025
Status: Ready for Vercel deployment
Tested: ✅ Local build successful
Security: ✅ All vulnerabilities resolved