🔐 Render Authentication Issue - Complete Fix Guide

📋 Problem Summary

Based on the screenshots and code analysis, the authentication issue on Render deployment is caused by:

1. **Frontend connecting to localhost** - The frontend is trying to connect to `http://localhost:3001` instead of the production backend URL
2. **Missing Environment Variable** - The `VITE_BACKEND_URL` environment variable is not properly set in Render
3. **CORS Configuration** - While CORS is configured, we've improved it to handle multiple origins better

✅ What We've Fixed

1. Improved CORS Configuration

- **File**: `backend/src/app.ts`
- **Changes**:
- Added support for comma-separated CORS_ORIGIN environment variable
- Added detailed logging for debugging CORS issues
- Included all frontend domains (Vercel + Render)

2. Created Render Configuration

- **File**: `render.yaml`
- **Purpose**: Automates deployment of both frontend and backend services
- **Features**:
- Proper build commands for both services
- Environment variables pre-configured
- Health check endpoints

3. Updated Environment Examples

- **Files**: `.env.example`, `backend/.env.example`
- **Changes**: Updated with production URLs and comma-separated CORS origins

🚀 Step-by-Step Fix Instructions

Step 1: Push Changes to GitHub

```
cd /home/ubuntu/workigom
git add .
git commit -m "Fix: Render authentication issue - Update CORS and environment config-
uration"
git push origin master
```

Step 2: Configure Backend on Render

2.1 Check Your Backend Service

1. Go to Render Dashboard (https://dashboard.render.com/)
2. Find your **workigom-backend** service
3. If it doesn't exist, create a new **Web Service**:
   - Connect your GitHub repository
   - Name: `workigom-backend`
   - Root Directory: Leave empty or set to `backend`
   - Environment: `Node`
   - Build Command: `npm install && npx prisma generate && npm run build`
   - Start Command: `npm run start`

2.2 Set Backend Environment Variables

In your backend service, go to **Environment** tab and add these variables:

Required Variables:

```
NODE_ENV=production
DATABASE_URL=<your-postgresql-connection-string>
JWT_SECRET=<generate-a-secure-random-string>
JWT_EXPIRES_IN=7d
JWT_REFRESH_SECRET=<generate-another-secure-random-string>
JWT_REFRESH_EXPIRES_IN=30d
CORS_ORIGIN=https://workigom.vercel.app,https://workigom-frontend1.onrender.com
```

Optional Variables:

```
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100
MAX_FILE_SIZE=5242880
ALLOWED_FILE_TYPES=image/jpeg,image/png,image/jpg,application/pdf
```

Generate Secure Secrets:

```
# Generate JWT_SECRET
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"

# Generate JWT_REFRESH_SECRET
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

2.3 Create PostgreSQL Database (if not exists)

1. In Render Dashboard, click **New +**
2. Select **PostgreSQL**
3. Name: `workigom-db`
4. Database: `workigom_db`
5. User: `workigom_user`
6. Region: Same as your backend (e.g., Frankfurt)
7. Plan: Free
8. Click **Create Database**
9. Copy the **Internal Database URL**

10. Add it as `DATABASE_URL` in your backend environment variables

2.4 Deploy Backend

1. Click **Manual Deploy** → **Deploy latest commit**
2. Wait for deployment to complete (3-5 minutes)
3. Check logs for any errors
4. Once deployed, your backend will be at: `https://workigom-backend.onrender.com`

2.5 Test Backend Health

```
curl https://workigom-backend.onrender.com/api/health
```

Expected response:

```
{
  "success": true,
  "message": "Workigom API is running",
  "database": "connected"
}
```

Step 3: Configure Frontend on Render

3.1 Check Your Frontend Service

1. Go to Render Dashboard (https://dashboard.render.com/)
2. Find your **workigom-frontend** service
3. If it doesn't exist, create a new **Static Site**:
   - Connect your GitHub repository
   - Name: `workigom-frontend`
   - Build Command: `npm install && npm run build:frontend`
   - Publish Directory: `dist`

3.2 Set Frontend Environment Variables

In your frontend service, go to **Environment** tab and add:

Critical Variable:

```
VITE_BACKEND_URL=https://workigom-backend.onrender.com
```

⚠️ **IMPORTANT**: Make sure there's NO `/api` suffix in the URL. The frontend code automatically adds it.

3.3 Deploy Frontend

1. Click **Manual Deploy** → **Clear build cache & deploy**
2. Wait for deployment to complete (2-3 minutes)
3. Once deployed, your frontend will be at: `https://workigom-frontend1.onrender.com`

Step 4: Alternative - Deploy on Vercel

If you prefer to use Vercel for the frontend:

4.1 Configure Vercel Project

1. Go to Vercel Dashboard (https://vercel.com/dashboard)

2. Import your GitHub repository
3. Configure:
   - Framework Preset: **Vite**
   - Root Directory: Leave empty
   - Build Command: `npm run build:frontend`
   - Output Directory: `dist`

4.2 Set Environment Variable

In Vercel project settings:
- **Environment Variables** → Add:
`VITE_BACKEND_URL=https://workigom-backend.onrender.com`
- Apply to: **Production, Preview, and Development**

4.3 Deploy

- Vercel will automatically deploy
- Your frontend will be at: `https://workigom.vercel.app`

Step 5: Verify Authentication

5.1 Open Browser Developer Tools

1. Open your frontend URL (Render or Vercel)
2. Press `F12` to open Developer Tools
3. Go to **Console** tab
4. Look for the API configuration log:
   ```
   🔧 API Configuration: {
     VITE_BACKEND_URL: "https://workigom-backend.onrender.com",
     finalApiUrl: "https://workigom-backend.onrender.com/api",
     mode: "production"
   }
   ```

5.2 Test Login

1. Go to Login page
2. Switch to **Network** tab in Developer Tools
3. Filter by `XHR` or `Fetch`
4. Try to login with test credentials:
   - **Email**: `test@example.com`
   - **Password**: `password123`
5. Check the login request:
   - URL should be: `https://workigom-backend.onrender.com/api/auth/login`
   - Method: `POST`
   - Status: Should be `200 OK` (not 401 or 404)
   - Response should include a `token`

5.3 Check for Errors

If you see CORS errors:
1. Check backend logs in Render dashboard
2. Look for messages like:
`❌ CORS: Blocked origin https://your-frontend-domain.com`
3. Add the blocked origin to `CORS_ORIGIN` environment variable in backend

If you see 401 Unauthorized:
1. Check if the backend database is seeded with test users
2. Run the seed endpoint: `POST https://workigom-backend.onrender.com/api/seed`
3. Try logging in again

If you see ERR_CONNECTION_REFUSED:
1. Backend is not running or not deployed
2. Check backend deployment status in Render
3. Verify backend URL is correct

🔍 Debugging Tips

Check Backend Logs

1. Go to Render Dashboard → Your backend service
2. Click **Logs** tab
3. Look for:
   ```
   🔒 CORS Allowed Origins: [...]
     ✅ CORS: Allowing origin https://...
     ❌ CORS: Blocked origin https://...
   ```

Check Frontend Console

1. Open browser DevTools → Console
2. Look for API configuration
3. Check for any error messages

Test Backend Directly

```
# Health check
curl https://workigom-backend.onrender.com/api/health

# Test login (replace with your test user)
curl -X POST https://workigom-backend.onrender.com/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"test@example.com","password":"password123"}'
```

📝 Environment Variables Checklist

Backend (Render)

- ✅ `NODE_ENV=production`
- ✅ `DATABASE_URL` (PostgreSQL connection string)
- ✅ `JWT_SECRET` (secure random string)
- ✅ `JWT_REFRESH_SECRET` (secure random string)
- ✅ `CORS_ORIGIN` (comma-separated frontend URLs)

Frontend (Render or Vercel)

- ✅ `VITE_BACKEND_URL` (backend URL without /api suffix)

🎯 Expected Results

After following all steps:

1. ✅ Backend is running at `https://workigom-backend.onrender.com`
2. ✅ Frontend is running at `https://workigom-frontend1.onrender.com` or `https://workigom.vercel.app`
3. ✅ Login request goes to the correct backend URL
4. ✅ No CORS errors in console
5. ✅ Authentication works and user is redirected to dashboard

🆘 Still Having Issues?

Common Issues and Solutions

Issue 1: Frontend still connecting to localhost

Solution:
1. Clear browser cache (Ctrl+Shift+Delete)
2. Hard refresh (Ctrl+F5)
3. Check if `VITE_BACKEND_URL` is set in Render/Vercel
4. Redeploy frontend with “Clear build cache”

Issue 2: CORS errors persist

Solution:
1. Check exact frontend URL in browser address bar
2. Add it to `CORS_ORIGIN` in backend environment variables
3. Redeploy backend
4. Format: `https://domain1.com,https://domain2.com` (no spaces)

Issue 3: 401 Unauthorized on login

Solution:
1. Seed the database: `POST /api/seed`
2. Check if user exists in database
3. Verify password is correct
4. Check backend logs for authentication errors

Issue 4: Backend deployment fails

Solution:
1. Check build logs in Render
2. Make sure `DATABASE_URL` is set
3. Verify PostgreSQL database is created
4. Check if `npx prisma generate` runs successfully

📚 Additional Resources

- Render Documentation (https://render.com/docs)
- Vite Environment Variables (https://vitejs.dev/guide/env-and-mode.html)
- Express CORS Middleware (https://expressjs.com/en/resources/middleware/cors.html)

✨ Summary

The authentication fix involved:
1. ✅ Improved CORS configuration with better logging
2. ✅ Created Render configuration file for automated deployment
3. ✅ Updated environment variable documentation
4. ✅ Added support for multiple frontend origins
5. ✅ Provided comprehensive deployment guide

Next Step: Follow the instructions above, starting with pushing changes to GitHub, then configuring your Render services with the correct environment variables.

Last Updated: November 1, 2025
Status: Ready to deploy ✅