AI Workbench Deployment Guide

Complete guide for deploying the AI Workbench React.js clone to various platforms.

🏗️ Build Process

1. Local Build

```
cd ai_workbench_clone/app
yarn install
yarn build
```

2. Static Export (for S3/GitHub Pages)

```
yarn export
```

This creates an `out` folder with static files.

🌐 Deployment Options

Option 1: AWS S3 Static Website

Step 1: Create S3 Bucket

1. Log into AWS Console
2. Go to S3 service
3. Create new bucket (e.g., `ai-workbench-app`)
4. Uncheck “Block all public access”
5. Acknowledge public access

Step 2: Configure Static Website Hosting

1. Go to bucket Properties
2. Scroll to “Static website hosting”
3. Click Edit
4. Enable static website hosting
5. Set index document: `index.html`
6. Set error document: `404.html`
7. Save changes

Step 3: Upload Files

1. Build your app: `yarn build && yarn export`
2. Upload all contents of `out` folder to S3 bucket root
3. Make sure all files are publicly readable

Step 4: Configure Bucket Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::ai-workbench-app/*"
        }
    ]
}
```

Step 5: Access Your App

- Website URL: `http://ai-workbench-app.s3-website-[region].amazonaws.com`

Option 2: GitHub Pages

Step 1: Push to GitHub

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin https://github.com/[username]/ai-workbench-clone.git
git push -u origin main
```

Step 2: Configure GitHub Pages

1. Go to repository Settings
2. Scroll to Pages section
3. Source: Deploy from a branch
4. Branch: main / (root)
5. Save

Step 3: Access Your App

- URL: `https://[username].github.io/ai-workbench-clone`

Option 3: Vercel (Recommended)

Step 1: Push to GitHub

(Same as Option 2, Step 1)

Step 2: Deploy to Vercel

1. Go to vercel.com (https://vercel.com)
2. Sign in with GitHub
3. Click “New Project”
4. Import your repository
5. Configure build settings:
   - Framework Preset: Next.js
   - Build Command: `yarn build`
   - Output Directory: `.next`

6. Deploy

Step 3: Access Your App

- Vercel provides a URL like: `https://ai-workbench-clone.vercel.app`

Option 4: Netlify

Step 1: Build Settings

Create `netlify.toml`:

```
[build]
  publish = "out"
  command = "yarn build && yarn export"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200
```

Step 2: Deploy

1. Go to netlify.com (https://netlify.com)
2. Connect GitHub repository
3. Configure build settings (or use netlify.toml)
4. Deploy

🔧 Configuration Files

For Static Export (S3/GitHub Pages)

Update `next.config.js`:

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  images: {
    unoptimized: true,
  },
  trailingSlash: true,
  output: 'export',
}

module.exports = nextConfig
```

For Dynamic Hosting (Vercel/Netlify)

Standard Next.js config works:

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  images: {
    domains: ['via.placeholder.com'],
  },
}

module.exports = nextConfig
```

📁 File Structure for Deployment

```
ai_workbench_clone/
├── app/                    # Main application code
├── out/                    # Static export (generated)
├── .gitignore              # Git ignore rules
├── README.md               # Main documentation
├── DEPLOYMENT.md           # This file
└── package.json            # Dependencies (if copying separately)
```

🚨 Common Issues & Solutions

Images Not Loading

- Ensure `next.config.js` has `unoptimized: true` for static exports
- Check image paths are correct
- Verify images are included in the build

Routing Issues

- For static exports, use trailing slashes
- Configure redirects for SPA behavior
- Ensure all routes have corresponding HTML files

Build Errors

- Check Node.js version (18+)
- Clear `.next` folder and rebuild
- Verify all dependencies are installed

CSS Not Applied

- Ensure CSS files are imported correctly
- Check build process includes CSS files
- Verify Tailwind CSS is configured properly

🔍 Testing Deployment

Local Testing

```
# Test static export locally
yarn build
yarn export
npx serve out

# Test production build
yarn build
yarn start
```

Validation Checklist

- [ ] All pages load correctly
- [ ] Navigation works between pages
- [ ] Images display properly

- [ ] Tutorial tour functions
- [ ] Responsive design works
- [ ] No console errors
- [ ] All interactive elements work

🎯 Performance Optimization

Before Deployment

1. **Optimize Images**: Compress images and use appropriate formats
2. **Code Splitting**: Ensure components are properly code-split
3. **Bundle Analysis**: Check bundle size with `yarn analyze`
4. **Caching**: Configure proper cache headers

After Deployment

1. **CDN Setup**: Use CloudFront (AWS) or similar CDN
2. **Compression**: Enable Gzip/Brotli compression
3. **Monitoring**: Set up performance monitoring
4. **SEO**: Add proper meta tags and sitemap

📊 Analytics & Monitoring

Add Google Analytics

1. Create GA4 property
2. Add tracking code to `app/layout.tsx`
3. Configure events for user interactions

Monitor Performance

1. Use Lighthouse for performance audits
2. Monitor Core Web Vitals
3. Set up error tracking (Sentry, etc.)

🔐 Security Considerations

Static Deployment Security

- No server-side vulnerabilities
- Client-side code is public
- Secure API keys (don’t expose in frontend)
- Use HTTPS (enforced by most platforms)

Content Security Policy

Add CSP headers for enhanced security:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self' 'unsafe-eval'; style-src 'self' 'unsafe-inline';">
```

📞 Support

For deployment issues:
1. Check platform-specific documentation
2. Verify configuration files
3. Test locally before deploying
4. Check browser console for errors
5. Review build logs for issues

Happy Deploying! 🚀