Airiam RAG Service Frontend - Deployment Documentation

Table of Contents

Azure Deployment Guide

Azure Static Web Apps Configuration

1. **Create an Azure Static Web App**
   - Sign in to the Azure Portal
   - Click "Create a resource" > "Web" > "Static Web App"
   - Fill in the following details:
     - **Subscription**: Select your Azure subscription
     - **Resource Group**: Create new or select existing
     - **Name**: `airiam-rag-frontend`
     - **Region**: Choose the region closest to your users
     - **SKU**: Select "Standard" for production environments or "Free" for testing
     - **Source**: GitHub (or your preferred source control)
     - **Organization**: Your GitHub organization
     - **Repository**: Select your RAG frontend repository
     - **Branch**: `main` (or your production branch)
   - Click "Review + Create" > "Create"
2. **Configure Build Settings**
   - In the Azure Portal, navigate to your newly created Static Web App
   - Go to "Configuration" > "Build"
   - Configure the following build settings:

- **App location**: `/` (or the relative path to your app root)
- **Api location**: `api` (if you have API functions)
- **Output location**: `.next` (for Next.js applications)
- **Build command**: `npm run build`

3. **Configure Next.js for Static Web Apps**
   - Ensure your `next.config.js` includes:

javascript Copy

```
module.exports = {
  output: 'standalone',
  // Other Next.js configuration
};
```

Environment Variable Setup

1. **Add Environment Variables in Azure Portal**
   - Navigate to your Static Web App in the Azure Portal
   - Go to "Configuration" > "Application settings"
   - Add the following environment variables:
     - `NEXT_PUBLIC_API_URL`: URL of your RAG backend API
     - `NEXT_PUBLIC_AUTH_DOMAIN`: Authentication domain
     - `NEXT_PUBLIC_TENANT_ID`: Default tenant ID (if applicable)
     - `NEXT_PUBLIC_STORAGE_URL`: URL for file storage
     - `NEXT_PUBLIC_OAUTH_PROVIDERS`: Comma-separated list of enabled OAuth providers (e.g., "google,microsoft,apple")

2. **Environment Variable Handling for Production Builds**
   - Create a `.env.production` file in your project root (do not commit to source control)
   - Add production-specific variables that will be used during the build process
   - For runtime environment variables, use Azure's application settings

CI/CD Pipeline Configuration

1. **GitHub Actions Workflow**
   - A workflow file is automatically created by Azure Static Web Apps

- Customize the workflow in `.github/workflows/azure-static-web-apps-*.yml`:

yaml

Copy

```yaml
name: Azure Static Web Apps CI/CD

on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened, closed]
    branches:
      - main

jobs:
  build_and_deploy_job:
    if: github.event_name == 'push' || (github.event_name == 'pull_request' && github.event
    runs-on: ubuntu-latest
    name: Build and Deploy Job
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'npm'
      - name: Install dependencies
        run: npm ci
      - name: Lint
        run: npm run lint
      - name: Type check
        run: npm run type-check
      - name: Build And Deploy
        id: builddeploy
        uses: Azure/static-web-apps-deploy@v1
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN }}
          repo_token: ${{ secrets.GITHUB_TOKEN }}
          app_location: "/"
          api_location: "api"
          output_location: ".next"
          app_build_command: "npm run build"
        env:
          NEXT_PUBLIC_API_URL: ${{ secrets.NEXT_PUBLIC_API_URL }}
```

```
          NEXT_PUBLIC_AUTH_DOMAIN: ${{ secrets.NEXT_PUBLIC_AUTH_DOMAIN }}
          NEXT_PUBLIC_TENANT_ID: ${{ secrets.NEXT_PUBLIC_TENANT_ID }}
          NEXT_PUBLIC_STORAGE_URL: ${{ secrets.NEXT_PUBLIC_STORAGE_URL }}
          NEXT_PUBLIC_OAUTH_PROVIDERS: ${{ secrets.NEXT_PUBLIC_OAUTH_PROVIDERS }}

  close_pull_request_job:
    if: github.event_name == 'pull_request' && github.event.action == 'closed'
    runs-on: ubuntu-latest
    name: Close Pull Request Job
    steps:
      - name: Close Pull Request
        id: closepullrequest
        uses: Azure/static-web-apps-deploy@v1
        with:
          azure_static_web_apps_api_token: ${{ secrets.AZURE_STATIC_WEB_APPS_API_TOKEN }}
          action: "close"
```

2. **Setting up GitHub Secrets**
   - In your GitHub repository, go to "Settings" > "Secrets and variables" > "Actions"
   - Add the following secrets:
     - `AZURE_STATIC_WEB_APPS_API_TOKEN`: Generated by Azure when you create the Static Web App
     - `NEXT_PUBLIC_API_URL`: Your backend API URL
     - `NEXT_PUBLIC_AUTH_DOMAIN`: Authentication domain
     - `NEXT_PUBLIC_TENANT_ID`: Default tenant ID
     - `NEXT_PUBLIC_STORAGE_URL`: Storage URL
     - `NEXT_PUBLIC_OAUTH_PROVIDERS`: Enabled OAuth providers

Mobile Testing and Optimization Settings

1. **Configure Responsive Design Testing**
   - Enable Azure's built-in preview environments for PRs to test mobile layouts
   - In the Azure Portal, go to your Static Web App > "Environments"
   - Each PR creates a unique staging environment for testing

2. **Mobile Performance Monitoring**
   - Enable Application Insights for your Static Web App:
     - Go to your Static Web App > "Monitoring" > "Application Insights"

- Click "Create new" or link to an existing Application Insights resource
- Enable "Client-side telemetry"

3. **Configure Custom Domains and HTTPS**
   - In the Azure Portal, navigate to your Static Web App
   - Go to "Custom domains" > "Add"
   - Follow the prompts to configure your domain
   - HTTPS is automatically configured

Local Windows 11 Development Setup

Prerequisites Installation

1. **Install Node.js and npm**
   - Download and install Node.js 18 LTS or newer from nodejs.org
   - Verify installation with:

Copy

```
node -v
npm -v
```

2. **Install Git**
   - Download and install from git-scm.com
   - Verify installation with:

Copy

```
git --version
```

3. **Install Visual Studio Code (recommended)**
   - Download and install from code.visualstudio.com
   - Recommended extensions:
     - ESLint
     - Prettier
     - Tailwind CSS IntelliSense
     - Next.js snippets

4. **Install Windows Terminal (optional but recommended)**

- Install from Microsoft Store or GitHub releases

Environment Configuration

1. **Clone the Repository**

Copy

```
git clone https://github.com/your-org/airiam-rag-frontend.git
cd airiam-rag-frontend
```

2. **Install Dependencies**

Copy

```
npm install
```

3. **Configure Environment Variables**
   - Create a `.env.local` file in the project root with:

Copy

```
NEXT_PUBLIC_API_URL=http://localhost:8000
NEXT_PUBLIC_AUTH_DOMAIN=localhost
NEXT_PUBLIC_TENANT_ID=local-tenant
NEXT_PUBLIC_STORAGE_URL=http://localhost:8001/storage
NEXT_PUBLIC_OAUTH_PROVIDERS=google,microsoft
```

4. **Start the Development Server**

Copy

```
npm run dev
```

   - The application will be available at http://localhost:3000

Mobile Device Testing Setup

1. **Using Windows Device Emulation**
   - In Chrome or Edge, open DevTools (F12)
   - Click the "Toggle device toolbar" button or press Ctrl+Shift+M
   - Select from predefined device presets or configure custom dimensions

2. **Testing on Actual Mobile Devices**

- Enable network discovery in Windows 11:
  - Go to Settings > Network & Internet > Sharing options
  - Enable network discovery and file sharing
- Find your computer's IP address:

Copy

```
ipconfig
```

- On your mobile device, connect to the same network and navigate to:

Copy

```
http://YOUR_COMPUTER_IP:3000
```

3. **Using Browser Sync for Multi-device Testing**
   - Install Browser Sync:

Copy

```
npm install -g browser-sync
```

   - Start Browser Sync proxy for your Next.js development server:

Copy

```
browser-sync start --proxy localhost:3000 --files "**/*"
```

   - Access the provided external URL on any device on the same network

Performance Testing Tools

1. **Lighthouse Integration**
   - Run Lighthouse audits via Chrome DevTools:
     - Open DevTools (F12)
     - Go to "Lighthouse" tab
     - Select "Mobile" device
     - Check Performance, Accessibility, Best Practices, and SEO
     - Click "Generate report"
2. **React Developer Tools**

- Install React Developer Tools for Chrome or Edge
- Enable "Highlight updates when components render" to identify excessive re-renders

3. **Performance Monitoring with Next.js**
   - Use the built-in Next.js analytics:

Copy

```
npm run build
npm run start
```

   - Navigate to `/analytics` in your browser (requires enabling in next.config.js)

CodeSandbox Setup Guide

Configuration Steps

1. **Create a New CodeSandbox**
   - Go to codesandbox.io
   - Click "Create Sandbox"
   - Select "Import Project" and enter your repository URL
   - Alternatively, choose "Next.js" template to start from scratch

2. **Configure Package.json**
   - Ensure your package.json includes the necessary dependencies:
     - Next.js
     - React
     - Redux Toolkit
     - TailwindCSS
     - Other project dependencies

3. **Setup Project Structure**
   - Ensure the following files are properly configured:
     - `next.config.js`
     - `tailwind.config.js`
     - `postcss.config.js`
     - `.eslintrc.json`

Environment Variable Setup

1. **Add Environment Variables in CodeSandbox**
   - Click on the gear icon (Settings)
   - Go to "Server Control Panel" > "Environment Variables"
   - Add the same environment variables as specified in the local setup section
   - Make sure to prefix client-side variables with `NEXT_PUBLIC_`
2. **Create a .env File (Alternative)**
   - Add a `.env` file to your sandbox root
   - Add your environment variables
   - Note: Be cautious with sensitive values as they may be visible to anyone with access to your sandbox

Mobile Preview Configuration

1. **Using CodeSandbox Preview Modes**
   - Click on the "Browser" tab in the preview pane
   - Use the responsive design controls to switch between device sizes
   - Test common breakpoints: 375px (small mobile), 768px (tablet), 1024px (laptop)
2. **Sharing Preview Link for Mobile Testing**
   - Click "Share" in the top-right corner
   - Copy the "Sandbox URL"
   - Open this URL on a mobile device to test the actual mobile experience
3. **Configure Viewport Settings**
   - Ensure your application's HTML includes proper viewport meta tags:

```html
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

Progressive Web App (PWA) Setup

Service Worker Configuration

1. **Install Required Packages**

Copy

```
npm install next-pwa
```

2. **Configure Next.js for PWA**
   - Update your `next.config.js`:

javascript

Copy

```
const withPWA = require('next-pwa')({
  dest: 'public',
  disable: process.env.NODE_ENV === 'development',
  register: true,
  skipWaiting: true
});

module.exports = withPWA({
  // Your existing Next.js config
  output: 'standalone'
});
```

3. **Create Custom Service Worker (Optional)**
   - Create a `worker/index.js` file for custom service worker logic:

javascript

Copy

```
self.addEventListener('install', (event) => {
  console.log('Service worker installed');
});

self.addEventListener('activate', (event) => {
  console.log('Service worker activated');
});

// Add custom cache strategies here
```

Manifest Setup

1. **Create Web App Manifest**
   - Create `public/manifest.json`:

json

Copy

```json
{
  "name": "Airiam RAG Service",
  "short_name": "Airiam",
  "description": "Retrieval-Augmented Generation AI Assistant",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#3b82f6",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }
  ]
}
```

2. **Add Icons for the PWA**
   - Create icons at various sizes (at minimum: 192x192 and 512x512)
   - Place them in the `public/icons/` directory
   - Ensure the paths match those in the manifest
3. **Add Manifest Link to Document Head**
   - In your `pages/_document.js` or `app/layout.js`:

javascript

Copy

```javascript
// In _document.js
import { Html, Head, Main, NextScript } from 'next/document';

export default function Document() {
  return (
    <Html lang="en">
      <Head>
        <link rel="manifest" href="/manifest.json" />
        <meta name="theme-color" content="#3b82f6" />
        <link rel="apple-touch-icon" href="/icons/icon-192x192.png" />
      </Head>
      <body>
        <Main />
        <NextScript />
      </body>
    </Html>
  );
}
```

Offline Capabilities Testing

1. **Configure Cache Strategies**
   - In your `next.config.js`, customize the PWA caching strategy:

javascript

Copy

```javascript
const withPWA = require('next-pwa')({
  dest: 'public',
  runtimeCaching: [
    {
      urlPattern: /^https:\/\/fonts\.(?:googleapis|gstatic)\.com\/(.*)/,
      handler: 'CacheFirst',
      options: {
        cacheName: 'google-fonts',
        expiration: {
          maxEntries: 30,
          maxAgeSeconds: 60 * 60 * 24 * 365 // 1 year
        }
      }
    },
    {
      urlPattern: /\.(?:eot|otf|ttc|ttf|woff|woff2|font.css)$/i,
      handler: 'StaleWhileRevalidate',
      options: {
        cacheName: 'static-font-assets',
        expiration: {
          maxEntries: 30,
          maxAgeSeconds: 60 * 60 * 24 * 365
        }
      }
    },
    {
      urlPattern: /\.(?:jpg|jpeg|gif|png|svg|ico|webp)$/i,
      handler: 'StaleWhileRevalidate',
      options: {
        cacheName: 'static-image-assets',
        expiration: {
          maxEntries: 64,
          maxAgeSeconds: 60 * 60 * 24 * 30
        }
      }
    },
    {
      urlPattern: /\/api\/.*$/i,
      handler: 'NetworkFirst',
      options: {
        cacheName: 'apis',
```

```
        expiration: {
          maxEntries: 16,
          maxAgeSeconds: 60 * 60
        },
        networkTimeoutSeconds: 10
      }
    },
    {
      urlPattern: /.*/i,
      handler: 'NetworkFirst',
      options: {
        cacheName: 'others',
        expiration: {
          maxEntries: 32,
          maxAgeSeconds: 60 * 60
        },
        networkTimeoutSeconds: 10
      }
    }
  ]
});
```

2. **Testing Offline Functionality**
   - In Chrome DevTools:
     - Go to the "Application" tab
     - In the left sidebar, under "Service Workers", check "Offline"
     - Refresh the page to see how the app behaves offline
   - Test critical functionality:
     - Previously viewed pages should load
     - Critical assets should be available
     - Appropriate error messages should display for unavailable content

3. **Implement Offline Fallback Pages**
   - Create `pages/offline.js` for a custom offline experience:

javascript

Copy

```javascript
export default function Offline() {
  return (
    <div className="flex flex-col items-center justify-center min-h-screen text-center p-4"
      <h1 className="text-2xl font-bold mb-4">You are offline</h1>
      <p className="mb-6">Please check your internet connection and try again.</p>
      <button
        onClick={() => window.location.reload()}
        className="px-4 py-2 bg-blue-500 text-white rounded hover:bg-blue-600"
      >
        Retry
      </button>
    </div>
  );
}
```

Mobile Installation Testing

1. **Testing on Android**
   - Open Chrome on an Android device
   - Navigate to your deployed application
   - If the PWA is configured correctly, Chrome will show an "Add to Home Screen" banner
   - Alternatively, use the menu button and select "Add to Home Screen"
   - Verify that the app launches as a standalone application without browser UI
2. **Testing on iOS**
   - Open Safari on an iOS device
   - Navigate to your deployed application
   - Tap the Share button
   - Select "Add to Home Screen"
   - Verify that the app icon appears on the home screen
   - Launch the app and confirm it opens in standalone mode
3. **Debugging Installation Issues**
   - Use Lighthouse to audit PWA features:
     - Run a Lighthouse audit with the PWA category enabled
     - Address any issues in the "Installable" and "PWA Optimized" sections

- Common issues:
  - Missing or incorrect manifest
  - Icons not available or incorrectly sized
  - Service worker not registered correctly
  - HTTPS not configured

Performance Optimization for Mobile Devices

Code Optimization Strategies

1. **Implement Code Splitting**
   - Use Next.js dynamic imports for route-based code splitting:

javascript Copy

```
import dynamic from 'next/dynamic';

const DynamicComponent = dynamic(() => import('../components/HeavyComponent'), {
  loading: () => <p>Loading...</p>,
  ssr: false // Use this for components that should only render client-side
});
```

2. **Optimize Redux Store**
   - Use Redux Toolkit's createEntityAdapter for efficient data normalization
   - Implement selective state subscriptions with useSelector
   - Consider RTK Query for data fetching and caching

3. **Implement Virtualized Lists**
   - For long lists of files or chat messages, use virtualization:

Copy

```
npm install react-window
```

javascript

Copy

```javascript
import { FixedSizeList } from 'react-window';

function VirtualizedFileList({ files }) {
  return (
    <FixedSizeList
      height={500}
      width="100%"
      itemCount={files.length}
      itemSize={60}
    >
      {({ index, style }) => (
        <div style={style}>
          <FileListItem file={files[index]} />
        </div>
      )}
    </FixedSizeList>
  );
}
```

Asset Optimization

1. **Image Optimization**
   - Use Next.js Image component for automatic optimization:

javascript

Copy

```javascript
import Image from 'next/image';

function Avatar({ user }) {
  return (
    <Image
      src={user.avatarUrl}
      alt={`${user.name}'s avatar`}
      width={40}
      height={40}
      placeholder="blur"
      blurDataURL="..."
    />
  );
}
```

2. **Font Optimization**
   - Use `next/font` for optimized font loading:

javascript Copy

```javascript
import { Inter } from 'next/font/google';

const inter = Inter({
  subsets: ['latin'],
  display: 'swap',
  variable: '--font-inter',
});

export default function RootLayout({ children }) {
  return (
    <html lang="en" className={inter.variable}>
      <body>{children}</body>
    </html>
  );
}
```

3. **Implement Critical CSS**
   - Extract and inline critical CSS with the `critters` webpack plugin
   - Add to your `next.config.js`:

javascript

Copy

```
module.exports = {
  webpack: (config, { isServer }) => {
    if (!isServer) {
      config.optimization.splitChunks.cacheGroups = {
        ...config.optimization.splitChunks.cacheGroups,
        styles: {
          name: 'styles',
          test: /\.(css|scss)$/,
          chunks: 'all',
          enforce: true,
        },
      };
    }
    return config;
  },
};
```

Mobile-Specific Optimizations

1. **Touch Optimization**
   - Increase touch target sizes for mobile:

css

Copy

```
@media (max-width: 640px) {
  .touch-target {
    min-height: 44px;
    min-width: 44px;
  }
}
```

2. **Reduce Motion for Accessibility**
   - Respect user preferences with reduced motion media query:

```css
@media (prefers-reduced-motion: reduce) {
  * {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
    scroll-behavior: auto !important;
  }
}
```

3. **Network-Aware Components**
   - Implement network detection and data saving modes:

```javascript
function useNetworkStatus() {
  const [isOnline, setIsOnline] = useState(true);
  const [saveData, setSaveData] = useState(false);

  useEffect(() => {
    setIsOnline(navigator.onLine);
    setSaveData(navigator.connection?.saveData || false);

    const handleOnline = () => setIsOnline(true);
    const handleOffline = () => setIsOnline(false);

    window.addEventListener('online', handleOnline);
    window.addEventListener('offline', handleOffline);

    return () => {
      window.removeEventListener('online', handleOnline);
      window.removeEventListener('offline', handleOffline);
    };
  }, []);

  return { isOnline, saveData };
}
```

Troubleshooting Common Issues

Deployment Issues

1. **Build Failures in CI/CD**
   - **Issue**: Build fails in GitHub Actions but works locally
   - **Solution**:
     - Verify Node.js version matches between local and CI environment
     - Check for environment variables that might be missing in CI
     - Ensure all dependencies are properly committed (package.json and lock file)

2. **Routing Issues After Deployment**
   - **Issue**: 404 errors when navigating to routes directly
   - **Solution**:
     - Configure Azure Static Web Apps routes in `staticwebapp.config.json`:

     json

     Copy

     ```
     {
       "navigationFallback": {
         "rewrite": "/index.html",
         "exclude": ["/images/*.{png,jpg,gif}", "/css/*"]
       }
     }
     ```

3. **Environment Variables Not Working**
   - **Issue**: Environment variables are undefined in the deployed app
   - **Solution**:
     - Ensure variables are prefixed with `NEXT_PUBLIC_` for client-side use
     - Verify variables are correctly set in Azure Portal
     - Check that variables are being injected during build time in CI/CD

Development Environment Issues

1. **Hot Reloading Not Working**
   - **Issue**: Changes don't reflect immediately during development
   - **Solution**:
     - Restart the development server
     - Check for syntax errors in the console
     - Verify that your file is being watched (not in .gitignore or excluded)

2. **TypeScript Errors**

- **Issue**: TypeScript errors preventing compilation
- **Solution**:
  - Run `npm run type-check` to see all errors
  - Update type definitions with `npm update @types/*`
  - Check for inconsistent Redux state types

3. **Package Dependencies Conflicts**
   - **Issue**: Incompatible dependencies causing build errors
   - **Solution**:
     - Clear npm cache: `npm cache clean --force`
     - Delete node_modules and reinstall: `rm -rf node_modules && npm install`
     - Check for peer dependency warnings and resolve them

Mobile-Specific Issues

1. **Touch Events Not Working Properly**
   - **Issue**: Click events not firing or delayed on mobile
   - **Solution**:
     - Replace click handlers with touch handlers for critical interactions
     - Eliminate hover effects that can cause delay (300ms tap delay)
     - Add the viewport meta tag with `user-scalable=no` for specific UI elements

2. **PWA Not Installing on Mobile**
   - **Issue**: "Add to Home Screen" prompt not appearing
   - **Solution**:
     - Verify manifest.json is correctly configured
     - Ensure you have icons of appropriate sizes (192x192 and 512x512 minimum)
     - Check that the service worker is registered correctly
     - Use Lighthouse to audit PWA capabilities

3. **Performance Issues on Low-end Devices**
   - **Issue**: App runs slowly or crashes on older mobile devices
   - **Solution**:
     - Implement device detection to serve simplified views to low-end devices
     - Reduce JavaScript bundle size through code splitting
     - Optimize renders by memoizing components and callbacks

- Consider implementing a "lite" mode that can be toggled by users