Deployment Guide - Outlook AI Assistant

This guide provides step-by-step instructions for deploying the Outlook AI Assistant to production environments.

Prerequisites

Before deploying, ensure you have:

1. **Azure AD App Registration** - Set up in Azure Portal
2. **AI Provider Account** - OpenAI, Azure OpenAI, or Anthropic
3. **Hosting Platform** - Vercel, Netlify, or custom server
4. **Domain Name** (optional but recommended for production)

Azure AD Configuration

Step 1: Create Azure AD App Registration

1. Go to Azure Portal (https://portal.azure.com/)
2. Navigate to **Azure Active Directory** > **App registrations**
3. Click **New registration**
4. Fill in the details:
   - **Name**: `Outlook AI Assistant`
   - **Supported account types**: `Accounts in any organizational directory and personal Microsoft accounts`
   - **Redirect URI**: `Web` - `https://your-domain.com/api/auth/callback/azure-ad`

Step 2: Configure App Settings

1. **Application (client) ID**: Copy this value for `AZURE_AD_CLIENT_ID`
2. **Directory (tenant) ID**: Copy this value for `AZURE_AD_TENANT_ID`
3. Go to **Certificates & secrets** > **New client secret**
4. Create a secret and copy the **Value** for `AZURE_AD_CLIENT_SECRET`

Step 3: Set API Permissions

1. Go to **API permissions**
2. Click **Add a permission** > **Microsoft Graph** > **Delegated permissions**
3. Add these permissions:
   - `openid`
   - `profile`
   - `email`
   - `User.Read`
   - `Mail.Read`
   - `Calendars.Read`
4. Click **Grant admin consent** (if you have admin rights)

Step 4: Configure Redirect URIs

Add these redirect URIs based on your deployment:

Development:
- `http://localhost:3000/api/auth/callback/azure-ad`

Production:
- `https://your-domain.com/api/auth/callback/azure-ad`

Environment Variables Setup

Required Variables

Create a `.env.local` file (or configure in your hosting platform):

```
# NextAuth.js Configuration
NEXTAUTH_SECRET=your-super-secret-jwt-key-here
NEXTAUTH_URL=https://your-domain.com

# Microsoft Azure AD Configuration
AZURE_AD_CLIENT_ID=your-client-id-from-azure
AZURE_AD_CLIENT_SECRET=your-client-secret-from-azure
AZURE_AD_TENANT_ID=your-tenant-id-from-azure

# AI Provider Configuration
AI_PROVIDER=openai
OPENAI_API_KEY=your-openai-api-key-here
OPENAI_MODEL=gpt-4
```

Generating NEXTAUTH_SECRET

Generate a secure secret for JWT signing:

```
# Option 1: Using OpenSSL
openssl rand -base64 32

# Option 2: Using Node.js
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Option 3: Online generator
# Visit: https://generate-secret.vercel.app/32
```

Deployment Platforms

Vercel (Recommended)

1. **Connect Repository:**
   `bash`
   `npm i -g vercel`
   `vercel login`
   `vercel --prod`

2. **Configure Environment Variables:**
   - Go to Vercel Dashboard > Your Project > Settings > Environment Variables
   - Add all required environment variables
   - Make sure `NEXTAUTH_URL` matches your Vercel domain

3. **Update Azure AD Redirect URI:**
   - Add your Vercel domain to Azure AD redirect URIs
   - Format: `https://your-app.vercel.app/api/auth/callback/azure-ad`

Netlify

1. **Build Settings:**
   ```bash
   # Build command
   npm run build

# Publish directory
.next
```

1. **Environment Variables:**
   - Go to Site Settings > Environment Variables
   - Add all required variables

2. **Netlify Configuration:**
   Create `netlify.toml`:
   ```toml
   [build]
   command = “npm run build”
   publish = “.next”

[[redirects]]
from = “/api/*”
to = “/.netlify/functions/:splat”
status = 200
```

Custom Server (Docker)

1. **Create Dockerfile:**
   ```dockerfile
   FROM node:18-alpine

WORKDIR /app
COPY package*.json ./
RUN npm ci –only=production

COPY . .
RUN npm run build

EXPOSE 3000
CMD [“npm”, “start”]
```

1. **Build and Run:**
   `bash`
   `docker build -t outlook-ai-assistant .`
   `docker run -p 3000:3000 --env-file .env.local outlook-ai-assistant`

Post-Deployment Checklist

1. Test Authentication Flow

1. Visit your deployed application
2. Click “Sign in with Microsoft”
3. Complete the OAuth flow
4. Verify you’re redirected to the dashboard
5. Test sign-out functionality

2. Test AI Chat Functionality

1. Navigate to the dashboard
2. Click the AI chat button
3. Send a test message
4. Verify AI responses are working

3. Verify Error Handling

1. Test with invalid credentials (if applicable)
2. Check error pages display correctly
3. Verify error messages are helpful

4. Performance Testing

1. Test page load speeds
2. Check mobile responsiveness
3. Verify animations work smoothly

Troubleshooting Common Issues

“Configuration Error” on Sign-in

Cause: Missing or incorrect environment variables

Solution:
1. Check all environment variables are set correctly
2. Verify `NEXTAUTH_SECRET` is generated and set
3. Ensure `NEXTAUTH_URL` matches your domain exactly
4. Restart your application after changing environment variables

“Access Denied” Error

Cause: Azure AD configuration issues

Solution:
1. Verify redirect URI in Azure AD matches your domain
2. Check API permissions are granted
3. Ensure the Azure AD app is configured for the correct account types

“OAuth Callback Error”

Cause: Redirect URI mismatch

Solution:
1. Check Azure AD redirect URI exactly matches: `https://your-domain.com/api/auth/callback/azure-ad`

2. Ensure no trailing slashes or extra characters
3. Verify the domain is accessible and SSL certificate is valid

AI Chat Not Working

Cause: AI provider configuration issues

Solution:
1. Verify AI provider API key is correct
2. Check API key has sufficient credits/quota
3. Ensure `AI_PROVIDER` environment variable is set correctly
4. Test API key directly with the provider's API

Build Failures

Cause: Missing dependencies or configuration

Solution:
1. Run `npm install` to ensure all dependencies are installed
2. Check for TypeScript errors: `npm run build`
3. Verify all environment variables are available during build
4. Check Node.js version compatibility (requires Node.js 18+)

Security Best Practices

Environment Variables

1. **Never commit `.env.local` to version control**
2. **Use different secrets for different environments**
3. **Rotate secrets regularly**
4. **Use your hosting platform's secure environment variable storage**

Azure AD Security

1. **Use least privilege principle** - only request necessary permissions
2. **Regularly review app permissions**
3. **Monitor sign-in logs** in Azure AD
4. **Set up conditional access policies** if needed

Application Security

1. **Keep dependencies updated**: `npm audit fix`
2. **Use HTTPS in production** (required for OAuth)
3. **Set up proper CORS policies**
4. **Monitor application logs** for suspicious activity

Monitoring and Maintenance

Health Checks

Set up monitoring for:
- Application uptime
- Authentication success rates

- AI API response times
- Error rates

Regular Maintenance

1. **Update dependencies monthly**
2. **Rotate API keys quarterly**
3. **Review Azure AD permissions annually**
4. **Monitor AI usage and costs**

Backup and Recovery

1. **Document all configuration settings**
2. **Keep backup of environment variables** (securely)
3. **Test disaster recovery procedures**
4. **Have rollback plan ready**

Support and Resources

Documentation Links

- NextAuth.js Documentation (https://next-auth.js.org/)
- Azure AD App Registration Guide (https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app)
- OpenAI API Documentation (https://platform.openai.com/docs)
- Vercel Deployment Guide (https://vercel.com/docs)

Getting Help

1. Check the application logs first
2. Review this troubleshooting guide
3. Test with a minimal configuration
4. Check Azure AD sign-in logs
5. Verify API provider status pages

Need help? Check the main README.md (./README.md) for additional configuration options and troubleshooting tips.