Microsoft Authentication Setup Guide

1. Register Your Application in Azure AD

1. Go to Azure Portal
2. Navigate to **Azure Active Directory → App registrations**
3. Click **New registration**
4. Fill in:
   - **Name**: Your App Name (e.g., "Internal Edge Tool")
   - **Supported account types**:
     - Choose "Single tenant" for your organization only
     - Choose "Multitenant" if you want any Microsoft account
   - **Redirect URI**:
     - Platform: Web
     - URI: `http://localhost:8000/auth/microsoft/callback` (for development)
     - Add your production URL later: `https://yourdomain.com/auth/microsoft/callback`
5. Click **Register**

2. Configure Your Application

Get Your IDs

1. On the app overview page, copy:
   - **Application (client) ID** → This is your `MICROSOFT_CLIENT_ID`
   - **Directory (tenant) ID** → This is your `MICROSOFT_TENANT_ID`

Create Client Secret

1. Go to **Certificates & secrets → Client secrets**
2. Click **New client secret**
3. Add description and expiration (recommend 24 months)
4. Copy the **Value** immediately → This is your `MICROSOFT_CLIENT_SECRET`
   - ⚠️ **Save this now!** You can't see it again

Set API Permissions (already configured by default)

1. Go to **API permissions**

2. You should see `User.Read` already added
3. This is sufficient for basic authentication

Configure Authentication

1. Go to **Authentication**
2. Under **Web** platform, ensure your redirect URIs are listed
3. Under **Implicit grant and hybrid flows**:
   - Check **ID tokens** (used for implicit and hybrid flows)
4. Under **Supported account types**, confirm your selection
5. Save changes

3. Update Your Environment Variables

Create or update your `.env` file:

```bash
# Microsoft Authentication
MICROSOFT_TENANT_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
MICROSOFT_CLIENT_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
MICROSOFT_CLIENT_SECRET=your-client-secret-value-here

# URLs
FRONTEND_URL=http://localhost:3000
BACKEND_URL=http://localhost:8000

# Session Configuration
SESSION_SECRET_KEY=your-random-session-secret-here
SESSION_LIFETIME_HOURS=24

# Keep your existing database and other settings
DATABASE_URL=sqlite:///./app.db
# ... other settings ...
```

4. Install Required Dependencies

```bash
```

```
pip install httpx authlib
# or if using MSAL
pip install msal
```

5. Update Your Database

Since your tables are empty, just run your application and it will create the new schema:

```bash
python main.py
```

6. Make the First User an Admin

After the first user logs in:

1. Access your database:

   ```bash
   sqlite3 app.db
   ```

2. Make the first user a superuser:

   ```sql
   UPDATE users SET is_superuser = 1 WHERE id = 1;
   UPDATE users SET roles = '["admin", "user"]' WHERE id = 1;
   ```

Or create a management script:

```python
```

```python
# scripts/make_admin.py
from backend.database import SessionLocal
from backend.models.auth_models import User

db = SessionLocal()
user = db.query(User).filter(User.email == "admin@company.com").first()
if user:
    user.is_superuser = True
    user.roles = ["admin", "user"]
    db.commit()
    print(f"Made {user.email} an admin")
else:
    print("User not found")
```

7. Frontend Integration

Your frontend needs to:

1. **Login Flow**:

```javascript
// Get login URL
const response = await fetch('/auth/microsoft/login');
const { auth_url } = await response.json();

// Redirect to Microsoft
window.location.href = auth_url;
```

2. **Handle Callback**:

```javascript
// After Microsoft redirects back with token in URL
const urlParams = new URLSearchParams(window.location.search);
const token = urlParams.get('token');

if (token) {
  // Store token
  localStorage.setItem('access_token', token);
  // Redirect to dashboard
  window.location.href = '/dashboard';
}
```

3. **API Calls**:

```javascript
// Include token in all API requests
fetch('/api/items', {
  headers: {
    'Authorization': `Bearer ${localStorage.getItem('access_token')}`
  }
});
```

8. Production Considerations

Security

1. Use HTTPS in production
2. Set secure session cookies
3. Implement CSRF protection properly
4. Consider storing sessions in Redis instead of database

Azure AD Settings

1. Add production redirect URIs
2. Set up proper token lifetimes
3. Configure company branding on login page
4. Set up Conditional Access policies if needed

Monitoring

1. Enable Azure AD sign-in logs
2. Set up alerts for failed authentications
3. Monitor session creation/usage

9. Testing

1. **Test Login**:
   - Navigate to `http://localhost:8000/auth/microsoft/login`
   - You should be redirected to Microsoft
   - After login, you should be redirected back with a token
2. **Test API Access**:

```bash
```

```
# Get your token from the redirect URL
TOKEN="your-token-here"

# Test authenticated endpoint
curl -H "Authorization: Bearer $TOKEN" http://localhost:8000/auth/me
```

3. **Test Logout**:

```
bash

curl -X POST -H "Authorization: Bearer $TOKEN" http://localhost:8000/auth/logout
```

Troubleshooting

"Invalid client" error

- Check your client ID and secret
- Ensure redirect URI matches exactly

"User not found" after login

- Check database connection
- Verify user was created in database

Token errors

- Check SESSION_SECRET_KEY is set
- Verify token hasn't expired

CORS errors

- Add your frontend URL to CORS_ORIGINS in settings