

- 1. Reference
 - 1.1. Next.js Authentication Made Easy with Microsoft Entra ID
 - 1.2. Next.js 16 Middleware DEPRECATED - Authentication In Proxy Or Data Access Layer?
 - 1.3. Overview of user consent and how to manage it in Microsoft Entra | Microsoft
 - 1.4. Add Tailwind CSS to an Existing Next js Project
 - 1.5. How to style Material UI component with Tailwindcss in React project
 - 1.6. How to Use Prisma 7 in Next.js
- 2. Project Structure
- 3. App Registrations
 - 3.1. Permissions Needed
- 4. Steps
 - 4.1. create nextjs app
 - 4.2. Create .env file
 - 4.3. Configure Git
 - 4.4. Test nextjs
 - 4.5. Check Authentication
 - 4.6. Install NextAuth.js v5
- 5. .env files
 - 5.1. .env.local
 - 5.2. .env
- 6. Prisma Install
 - 6.1. The CLI is for development tasks like migrations
 - 6.2. Create the prisma folder
- 7. Run this whenever you change your schema.prisma file
 - 7.1. Pro-Tip: Update your package.json
- 8. Where we are up to 8/1/2026
- 9. Testing Creating And Uploading Folders and Files
- 10. 📖 WebCalibre Development Notes 17/1/2026
 - 10.1. 🚀 Project Overview
 - 10.2. 🛠 Critical Technical Learnings
 - 10.2.1. MUI X v8 DataGrid Toolbar Fix
 - 10.2.2. Global "Hidden" Search
 - 10.2.3. Server-Side Extraction
 - 10.3. 📄 Essential Commands
 - 10.4. 📅 Future Roadmap
 - 10.4.1. Phase 1: Metadata Utility (In Progress)
 - 10.4.2. Phase 2: UI Enhancements
 - 10.5. 💡 Reminders

1. Reference

1.1. Next.js Authentication Made Easy with Microsoft Entra ID

[Next.js Authentication Made Easy with Microsoft Entra ID](#)

1.2. Next.js 16 Middleware DEPRECATED - Authentication In Proxy Or Data Access Layer?

[Next.js 16 Middleware DEPRECATED - Authentication In Proxy Or Data Access Layer?](#)

1.3. Overview of user consent and how to manage it in Microsoft Entra | Microsoft

[Overview of user consent and how to manage it in Microsoft Entra | Microsoft](#)

1.4. Add Tailwind CSS to an Existing Next js Project

[Add Tailwind CSS to an Existing Next js Project](#)

1.5. How to style Material UI component with Tailwindcss in React project

[How to style Material UI component with Tailwindcss in React project](#)

1.6. How to Use Prisma 7 in Next.js

[How to Use Prisma 7 in Next.js](#)

2. Project Structure

The following is the desire structure

```
web-calibre/
├── src/
│   ├── app/
│   │   ├── api/
│   │   │   ├── auth/
│   │   │   └── upload/
│   │   ├── dashboard/
│   │   ├── layout.tsx
│   │   └── page.tsx
│   ├── components/
│   ├── lib/
│   │   ├── prisma.ts
│   │   └── theme.ts
│   └── middleware.ts
├── prisma/
└── .env
```

All Routes (Pages & APIs)
API ROUTES
NextAuth routes [...nextauth]
Custom API for OneDrive sessions
Protected user area
Global Layout
Landing Page
MUI Components (FileTable, Navbar)
Config (Prisma, Theme, Auth)

<-- MUI Theme Defined Here
<-- Protected Routes Logic
DB Schema
Secrets

3. App Registrations

WebCalibre2 Application (client) ID = 'b549931a-f491-436b-b7bc-d37d8ca3c17e'

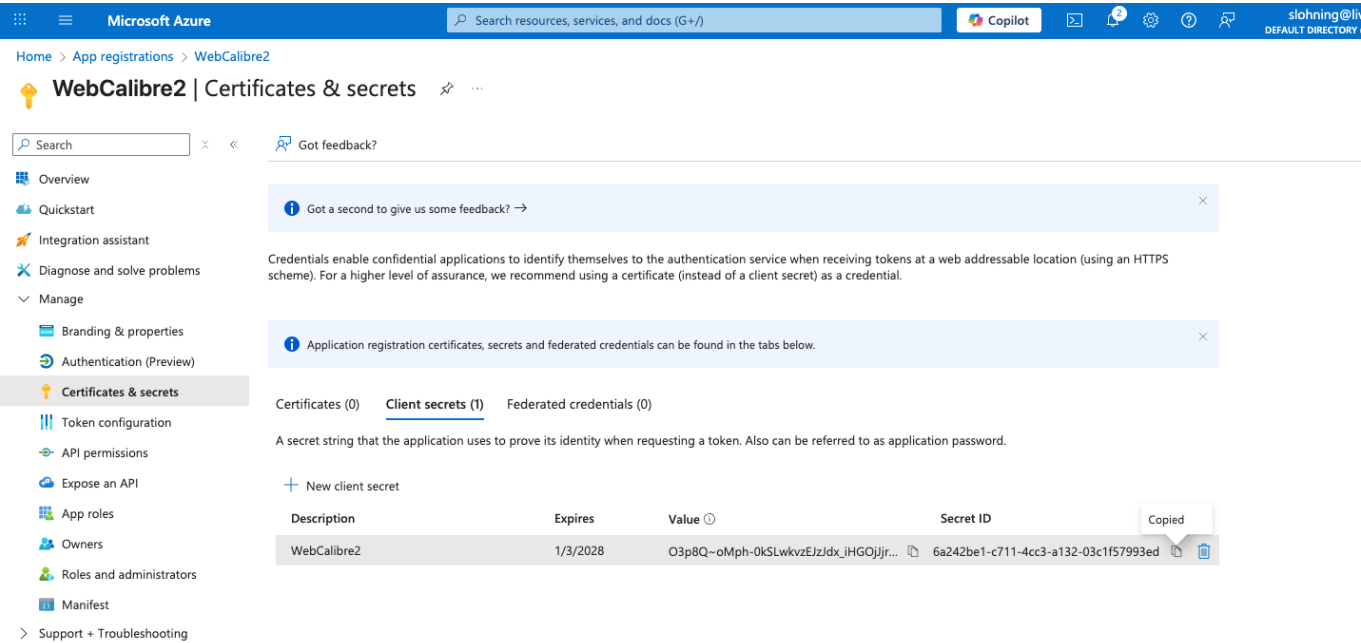
Object ID ='0b8256a2-e2ac-472b-896a-4b5845bc32fe'

Directory (tenant) ID = '1c06ce7c-7884-4796-8652-d4c32d75a5d0'

Expires = 1/3/2028

Value='O3p8Q~oMph-0kSLwkzEJzJdx_iHGOjJrr5Pa1A'

Secret ID='6a242be1-c711-4cc3-a132-03c1f57993ed'



Since you are using NextAuth.js, the URL must follow a very specific pattern.

For local development: `http://localhost:3000/api/auth/callback/azure-ad`

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

Note: Replace `azure-ad` with whatever ID you give the provider in your code. By default, in NextAuth, it is usually `azure-ad`.

WebCalibre3

Display name	WebCalibre3
Application (client) ID	a95ef644-cb9d-465c-8dae-f822a78a8ac3
Object ID	1e1f2bda-661c-4ad0-9aae-82313c603076
Directory (tenant) ID.	1c06ce7c-7884-4796-8652-d4c32d75a5d0

Client credentials

Description	Expires	Value	Secret ID
WebCalibre3	1/5/2028	VZ48Q~EOsgZcgBI25cc0zys.h9zM7hN9I7DhZdnW	d647cf28-80ed-45fb-a89e-e2bc323a40e9

3.1. Permissions Needed

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission ✓ Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin consent requ...	Status
Microsoft Graph (3) ...				
Files.ReadWrite	Delegated	Have full access to user files	No	...
offline_access	Delegated	Maintain access to data you have given it access to	No	✓ Granted for Default Dire... ...
User.Read	Delegated	Sign in and read user profile	No	✓ Granted for Default Dire... ...

Other permissions granted for Default Directory

4. Steps

4.1. create nextjs app

The following command creates the nextjs folder structure

Yes, Next.js has a specific syntax to create a project in your current directory instead of creating a new subfolder.

To do this, navigate to your empty project folder in the terminal and run:

```
npx create-next-app@latest .
```

Important Details:

1. The Dot (.): This tells the installer to use the current directory as the project root.
2. Naming Conflict: If the folder name contains capital letters (e.g., WebCalibre), the command will fail with an error. Next.js requires the root folder name to be all lowercase and URL-friendly. You must rename your folder to web-calibre before running the command. This caused me rename to all lower case.
3. Empty Folder: The directory must be empty (except for hidden files like .git). If you already have files in there, the installer will stop to prevent overwriting your work.

What happens next? Even though you are using the current directory, the interactive prompt will still ask you:

"What is your project named?" You can just hit Enter to accept the current folder name.

Configuration: It will then ask you about TypeScript, ESLint, Tailwind CSS, the src/ directory, and the App Router.

Based on our previous discussions, I recommend selecting "Yes" for TypeScript, ESLint, Tailwind, the src/ directory, and the App Router to match the professional structure we planned for WebCalibre.

4.2. Create .env file

In Next.js, the distinction between .env and .env.local is primarily about security and **overriding defaults**. While both store key-value pairs, they are handled differently by Git and the Next.js runtime.

1. **Security & Version Control (Git)** The most important reason is that .env.local is automatically ignored by Git in standard Next.js projects.
 - .env: Intended for "safe" default values that you want to share with your entire team. It is committed to your repository so that every developer has the same baseline configuration.
 - .env.local: Intended for secrets (API keys, database passwords, Azure secrets). Because it is not committed to Git, your private credentials never end up on GitHub or in your production code history.
2. **Priority & Overriding** Next.js uses a "hierarchy" of loading.¹ If you define the same variable in both files, .env.local wins. This allows you to set a generic placeholder in .env for the team, but use your specific local developer settings on your own machine. For example:
 - In .env: DATABASE_URL=postgres://localhost:5432/mydb (The team's default)
 - In .env.local: DATABASE_URL=postgres://admin:password123@localhost:5432/my_private_dev_db (Your personal setup)

The Next.js Loading Order Next.js looks for variables in this specific order (from highest priority to lowest):

1. process.env (System environment variables)
2. .env.development.local (Only during npm run dev)
3. .env.local (Available in all environments except test)
4. .env.development
5. .env (The final fallback)

```
DATABASE_URL="postgresql://stephen:Web2025$$@192.168.1.210:5432/webcalibre2"
```

```
# Credentials for your Microsoft App (NextAuth)
AZURE_AD_CLIENT_ID="549931a-f491-436b-b7bc-d37d8ca3c17e"
AZURE_AD_CLIENT_SECRET="6a242be1-c711-4cc3-a132-03c1f57993ed"
AZURE_AD_TENANT_ID="1c06ce7c-7884-4796-8652-d4c32d75a5d0"
```

Summary Comparison Table

Feature	.env	.env.local
Purpose	Shared defaults/constants	Private secrets & local overrides
Committed to Git?	Yes	No (Added to .gitignore)
Sensitive Data?	NEVER	YES (API Keys, Secrets)
Scope	All developers/environments	Just your machine

****Best Practice Tip: **** Since .env.local isn't shared on GitHub, it's a good idea to create a file named .env.example in your project. This file should contain the names of the keys (e.g., AZURE_AD_CLIENT_SECRET=) but leave the values blank, so other developers know which variables they need to create on their own machines.

4.3. Configure Git

Nextjs initializes git by default. But it doesn't add some configs that I do

```
git config --global push.followTags true
```

```
git remote add origin "/Users/stephenlohning/Library/CloudStorage/OneDrive-  
Personal/Documents/10_GIT_Repositories/124_WebCalibre2.git"
```

```
git remote -v origin /Users/stephenlohning/Library/CloudStorage/OneDrive-  
Personal/Documents/10_GIT_Repositories/124_WebCalibre2.git (fetch) origin  
/Users/stephenlohning/Library/CloudStorage/OneDrive-  
Personal/Documents/10_GIT_Repositories/124_WebCalibre2.git (push)
```

4.4. Test nextjs

```
npm run dev  
  
> 124_webcalibre2@0.1.0 dev  
> next dev  
  
▲ Next.js 16.1.1 (Turbopack)  
- Local:      http://localhost:3000  
- Network:    http://192.168.1.100:3000  
- Environments: .env  
  
✓ Starting...  
✓ Ready in 648ms
```

If it working proceed

4.5. Check Authentication

I ran through the Video:-

[Next.js Authentication Made Easy with Microsoft Entra ID](#)

4.6. Install NextAuth.js v5

```
npm install next-auth@beta
```

Setup Environment The only environment variable that is mandatory is the AUTH_SECRET. This is a random value used by the library to encrypt tokens and email verification hashes. (See Deployment to learn more). You can generate one via the official Auth.js CLI running:

npx auth secret

created .env.local

```
AUTH_SECRET="7cz3Z4kUI2kB3mdAPo58iioUDLSRJ92X8+boEixv08k=" # Added by `npx auth`. Read more: https://cli.authjs.dev
```

5. .env files

5.1. .env.local

```
# values generated by Gemini
# Generated for security
AUTH_SECRET="7cz3Z4kUI2kB3mdAPo58iioUDLSRJ92X8+boEixv08k="

# Use 'common' for multi-tenant (Work + Personal) support
AUTH_MICROSOFT_ENTRA_ID_TENANT_ID="common"

# The Application (client) ID
AUTH_MICROSOFT_ENTRA_ID_ID="b549931a-f491-436b-b7bc-d37d8ca3c17e"

# The Client Secret VALUE (ensure no leading '6' or spaces)
AUTH_MICROSOFT_ENTRA_ID_SECRET="03p8Q~oMph-0kSLwkvzEJzJdx_iHG0jJjrr5Pa1A"

# Required to tell Auth.js to trust your localhost/proxy URL
AUTH_TRUST_HOST=true
```

5.2. .env

```
DATABASE_URL="postgresql://stephen:Web2025$$$@192.168.1.210:5432/webcalibre2"

# Credentials for your Microsoft App (NextAuth)
AZURE_AD_CLIENT_ID="549931a-f491-436b-b7bc-d37d8ca3c17e"
AZURE_AD_CLIENT_SECRET="6a242be1-c711-4cc3-a132-03c1f57993ed"
AZURE_AD_TENANT_ID="1c06ce7c-7884-4796-8652-d4c32d75a5d0"
```

6. Prisma Install

To get Prisma fully installed and ready to talk to your PostgreSQL database, you need to install the CLI (for migrations) and the Client (for your code to query the database).

Run these three commands in your terminal:

1. Install the Prisma packages Bash

6.1. The CLI is for development tasks like migrations

```
npm install prisma --save-dev
```

##The Client is used by your Next.js code to talk to the DB

```
npm install @prisma/client
```

2. Install the Dotenv helper Since you are keeping your DATABASE_URL in .env.local (and your password has special characters), you need this to bridge the gap between Next.js and the Prisma CLI:

```
npm install dotenv-cli --save-dev
```

3. Initialize and Generate Once the packages are installed, you need to initialize the Prisma folder (if you haven't already) and generate the types for your FileNode model:

Bash

6.2. Create the prisma folder

```
npx prisma init
```

7. Run this whenever you change your schema.prisma file

`npx dotenv -e .env.local -- npx prisma generate` 4. Apply your Schema to PostgreSQL Finally, run this to actually create the tables in your webcalibre2 database:

```
npx dotenv -e .env.local -- npx prisma migrate dev --name init_database
```

7.1. Pro-Tip: Update your package.json

To avoid typing that long dotenv command every time, open your package.json and add this to the "scripts" section:

JSON

```
"scripts": { "db:migrate": "dotenv -e .env.local -- prisma migrate dev", "db:studio": "dotenv -e .env.local -- prisma studio" }
```

 Now, you can just run `npm run db:migrate` whenever you update your schema!

8. Where we are up to 8/1/2026

We can authenticate with MS Azure, upload a file and store it on my personal OneDrive. I can see a few problems,

1. if the file gets stored with its original filename, if I store the same file again it tells that it fail, but in reality it did not fail the it got renamed with 1.pdf added this is not added data base.
2. We may be better to create a folder and put the file in the folder, to me the logical name would id of the file, but that is automatically create and assigned when the data is stored as a UUID.
3. We would have to create the UUID ourselves, so that we can create a folder with that specific UUID and store the file in the folder.
4. The Upload file Page needs to be able to create folder as well select a parent , add a description of either the file

9. Testing Creating And Uploading Folders and Files

The functionality works may not the best user interface

Projects ID 0371fdfe-a2d1-4f25-b229-804f299bc064 Project-1 ID c8eebe0e-6e90-4cfb-8e44-c05f7064f3b1 parentID 0371fdfe-a2d1-4f25-b229-804f299bc064 AS-NZS3000-2018.pdf parentID c8eebe0e-6e90-4cfb-8e44-c05f7064f3b1

Now we will go now and add metadata

Now we have the metadata along with some great filters on the data board page

Now we are going to implement an an update of a file or folder stored. 12/1/2026

For download there 2 function we need to implement

1. down load the file to ~/user/Downloads, this should work for any file type including pdf
2. if the file is a pdf open the file in another tab in the browser as most browser support reading a pdf

For download there 2 function we need to implement

1. down load the file to ~/user/Downloads, this should work for any file type including pdf
2. if the file is a pdf open the file in another tab in the browser as most browser support reading a pdf

So we need to implement 2 new actions, downLoad file and open pdf to read. Does this make sense ?

10. WebCalibre Development Notes 17/1/2026

10.1. Project Overview

A Next.js (App Router) dashboard for managing a OneDrive-synced file library. Uses **Prisma + PostgreSQL** for metadata storage and **MUI X v8 (DataGrid)** for the frontend.

10.2. Critical Technical Learnings

10.2.1. MUI X v8 DataGrid Toolbar Fix

- **Problem:** The Search Bar and "Library" title often disappear even when defined in `slots`.
- **Solution:** - You MUST include the `showToolbar` prop on the `<DataGrid />`.
 - Use the **Persistent Quick Filter** model: set the `expanded` prop on the `<QuickFilter />` component within the custom toolbar to ensure it doesn't collapse.
 - Use `slotProps.textField` (not `input`) to customize the search input in MUI v7/v8.

10.2.2. Global "Hidden" Search

- To allow the search bar to find "Author", "ISBN", or "Dimensions" without cluttering the UI:
 - Create a column (e.g., `metadata_search`) with `width: 0`.
 - Use a `valueGetter` to return `JSON.stringify(row.metadata)`.
 - Hide the column in `initialState.columns.columnVisibilityModel`.
 - The DataGrid filtering engine will now index this hidden string.

10.2.3. Server-Side Extraction

- Metadata extraction from file binaries (PDF, EPUB, Images) **must** occur on the server.
- Libraries like `sharp` and `pdf-parse` are Node.js-based and will crash if imported into client components.

10.3. Essential Commands

Command	Purpose
<code>npx tsc --noEmit</code>	Run this frequently. Validates types across the entire project. Finds errors your IDE might miss.
<code>npx prisma generate</code>	Updates the Prisma Client types after schema changes.
<code>npx prisma db push</code>	Pushes schema changes to the PostgreSQL database.
<code>npm run dev</code>	Starts the development server.

10.4. Future Roadmap

10.4.1. Phase 1: Metadata Utility (In Progress)

- ☐ Implement `src/lib/metadata-extractor.ts`.
- ☐ Integrate extraction into the `syncOneDrive` Server Action.

- ☐ Supported formats: PDF (`pdf-parse`), EPUB (`node-epub-utils`), Images (`sharp`).

10.4.2. Phase 2: UI Enhancements

- ☐ Add "Last Synced" timestamp state to the dashboard header.
 - ☐ Add dedicated sortable columns for "Author" and "File Type".
 - ☐ Implement folder-specific row styling.
-

10.5. 💡 Reminders

- **Hard Refresh:** If the DataGrid UI behaves weirdly after code changes, use `Cmd + Shift + R` or `Ctrl + F5`.
- **Z-Index:** The search bar animation uses `gridArea: '1 / 1'`. If icons overlap, check the styled-components logic in `dashboard-view.tsx`.