National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

Sreeram Krishna
Student ID: X23292431

School of Computing
National College of Ireland

Supervisor: Prof. Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sreeram Krishna
Student ID:	X23292431
Programme:	Msc Cybersecurity **Year:** 2024-2025
Module:	Msc Research Project
Lecturer:	Prof. Michael Pantridge
Submission Due Date:	11-08-2025
Project Title:	Policy-Driven Identity and Access Management: Strengthening Compliance with Security Standards and Emerging Technologies
Word Count:	**1209** **Page Count: 9**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sreeram Krishna

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Azure AD Authentication with Next.js Configuration Manual

SREERAM KRISHNA
X23292431

1 Azure AD Setup (Azure Portal)

Register and configure your application within Azure Active Directory to enable Single Sign-On (SSO).

1.1 App Registration

Steps include registering the app, defining redirect URIs, and recording client credentials.

- Log in: https://portal.azure.com
- Azure Active Directory → App registrations → New registration
- Register Your App:
  - Name: Next.js Azure AD App
  - Supported account types: Single tenant
  - Redirect URI (Web): http://localhost:3000/api/auth/callback/azure-ad
- Click Register

Note Application (client) ID and Directory (tenant) ID for your .env.local.

1.2 Certificates & Secrets

Generate a new client secret and copy the value securely.

- Certificates & secrets → New client secret
- Description: NextAuth Secret, Expiry: 24 months
- Click Add and COPY the value immediately.

1.3 API Permissions

Set Microsoft Graph permissions including User.Read, GroupMember.Read.All, etc.

- API permissions → Add a permission → Microsoft Graph → Delegated permissions
- Add: User.Read (default), GroupMember.Read.All, Directory.Read.All (optional)
- Click Add permissions & Grant admin consent.

1.4 Authentication Settings

Enable ID tokens and define production/local redirect URIs.

- Redirect URIs: http://localhost:3000/api/auth/callback/azure-ad
- Production: https://yourdomain.com/api/auth/callback/azure-ad
- Enable ID tokens under Implicit grant and hybrid flows.
- Click Save

1.5 AD Groups (Optional)

Create groups like Admin Group for access control.

- Groups → New group → Security → Group name: Admin Group → Create
- Add members: Select users for admin rights.

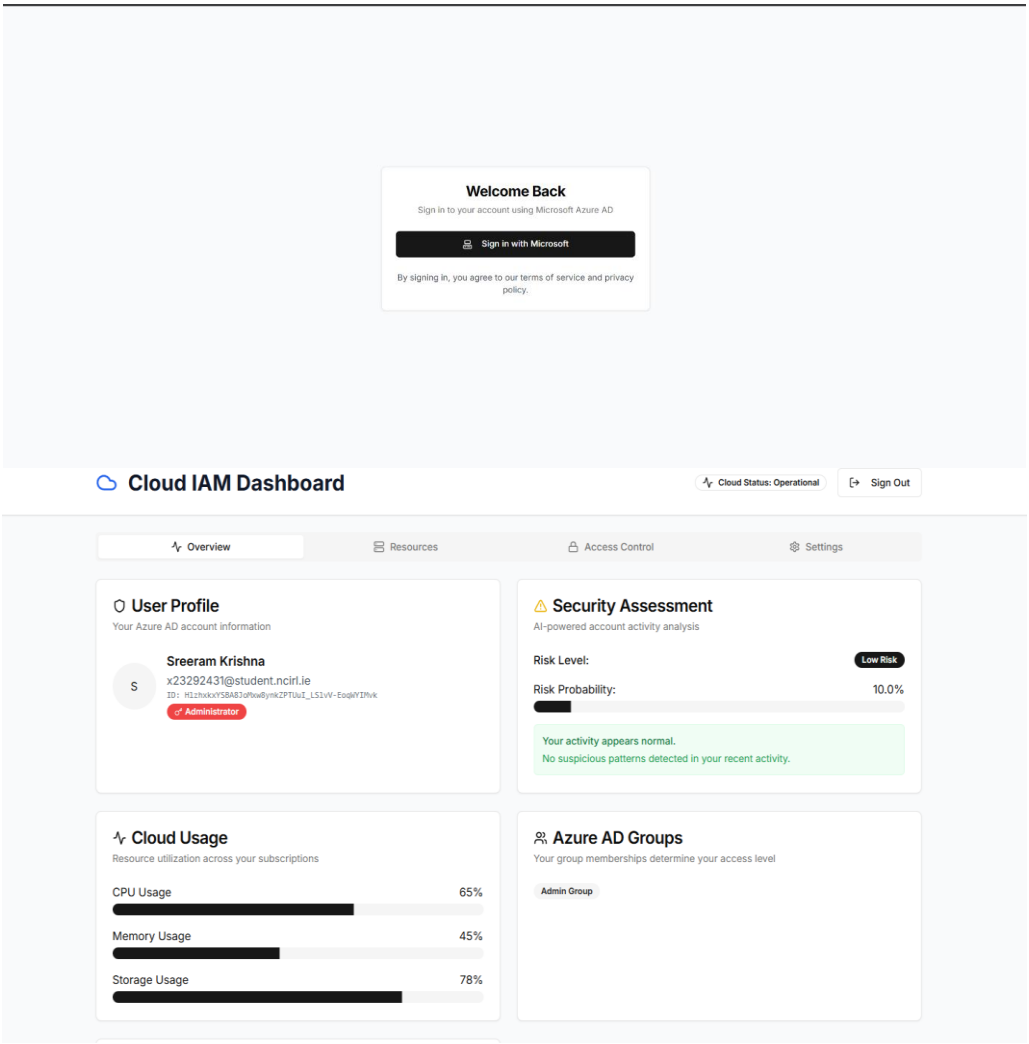

Figure 1. Microsoft Azure AD Authentication

2 Next.js App Initialization

Create and configure a new Next.js project:

- Initialize the Next.js project and install dependencies.

```
npx create-next-app@latest azure-ad-auth --typescript --tailwind --eslint --app
cd azure-ad-auth
```

- Install packages:

```
npm install next-auth @next-auth/prisma-adapter prisma @prisma/client
npm install @types/node
```

- Optional: Install UI libraries for admin panel.

```
npm install @radix-ui/react-avatar @radix-ui/react-badge ...
```

3 Environment Configuration

Create. env.local in project root:

```
AZURE_AD_CLIENT_ID=xxxxxxx
AZURE_AD_CLIENT_SECRET=xxxxxxx
AZURE_AD_TENANT_ID=xxxxxxx
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=xxxxx
DATABASE_URL="postgresql://postgres:password@localhost:5432/azure_auth_db"
```

Replace placeholders! For production, use Vercel/host environment variables.

4 Database & Prisma Setup

- Create a PostgreSQL database and user.
- Define schema in schema.prisma.
- Push the schema to DB and generate client:

Install PostgreSQL, create database and user:

```
CREATE DATABASE azure_auth_db;
CREATE USER azure_user WITH PASSWORD 'your_password';
GRANT ALL PRIVILEGES ON DATABASE azure_auth_db TO azure_user;
```

Initialize Prisma:

```
npx prisma init
```

Update prisma/schema.prisma as shown:

```
model User {
  id       String    @id @default(cuid())
  email    String    @unique
  name     String?
  image    String?
  azureId  String?   @unique
  groups   String[]
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt
  userActivities UserActivity[]
}

model UserActivity {
  id       String   @id @default(cuid())
  type     String
  uidType  String
  uid      String
  params   Json?
  role     String
  resource String
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  userId   String?
  user     User?    @relation(fields: [userId], references: [id], onDelete: SetNull)
```

```
  @@index([uid])
  @@index([type])
  @@index([role])
  @@index([resource])
  @@index([createdAt])
  @@index([userId])
}

npx prisma generate
npx prisma db push
```

5 Authentication & API Configuration

Configure NextAuth with AzureAD provider and Prisma adapter.
Location: app/api/auth/[...nextauth]/route.ts
Includes:

- Provider config
- JWT and session callbacks
- Group retrieval via Microsoft Graph API

```
import NextAuth from "next-auth"
import AzureADProvider from "next-auth/providers/azure-ad"
import { PrismaAdapter } from "@next-auth/prisma-adapter"
import { PrismaClient } from "@prisma/client"

const prisma = new PrismaClient()
export const authOptions = {
  providers: [
    AzureADProvider({
      clientId: process.env.AZURE_AD_CLIENT_ID!,
      clientSecret: process.env.AZURE_AD_CLIENT_SECRET!,
      tenantId: process.env.AZURE_AD_TENANT_ID!,
      authorization: {
        params: {
          scope: "openid profile email User.Read GroupMember.Read.All"
        }
      }
    })
  ],
  adapter: PrismaAdapter(prisma),
  session: { strategy: "jwt" },
  pages: { signIn: "/auth/signin", error: "/auth/error" },
  callbacks: {
    async jwt({ token, account, profile }) {
      if (account && profile) {
        token.accessToken = account.access_token
        token.groups = await fetchUserGroups(account.access_token!)
        token.id = profile.sub || profile.oid || account.providerAccountId
      }
      return token
    },
    async session({ session, token }) {
      session.accessToken = token.accessToken as string
      session.user.groups = token.groups as string[]
      session.user.id = token.id as string
      return session
    }
  }
}

async function fetchUserGroups(accessToken: string): Promise<string[]> {
  try {
```

```
    const res = await fetch("https://graph.microsoft.com/v1.0/me/memberOf", {
      headers: {
        Authorization: `Bearer ${accessToken}`,
        "Content-Type": "application/json"
      }
    })
    if (!res.ok) return []
    const data = await res.json()
    return data.value.map((group: any) => group.displayName).filter(Boolean)
  } catch {
    return []
  }
}

const handler = NextAuth(authOptions)
export { handler as GET, handler as POST }
```

6 Middleware & Access Control

Create middleware.ts at root to enforce group-based protection:

```
import { withAuth } from "next-auth/middleware"

export default withAuth(
  function middleware(req) {},
  {
    callbacks: {
      authorized: ({ token, req }) => {
        if (req.nextUrl.pathname.startsWith("/admin")) {
          return token?.groups?.includes("Admin Group") ?? false
        }
        if (req.nextUrl.pathname.startsWith("/dashboard")) {
          return !!token
        }
        return true
      }
    }
  }
)
export const config = {
  matcher: ["/dashboard/:path*", "/admin/:path*", "/profile/:path*"]
}
```

7 Running Locally & Testing

Steps:

1. Push DB schema: npx prisma db push
   ```
   npx prisma db push
   npx prisma generate
   ```
2. Start dev server: npm run dev
   ```
   npm run dev
   ```
3. Visit http://localhost:3000, sign in, and test routes.

8 Production Deployment

- Update Azure AD with production redirect URI. Add https://yourdomain.com/api/auth/callback/azure-ad

- Set environment variables in Vercel. (Project > Settings > Environment Variables)
- Deploy:

```
npm i -g vercel
vercel
```

9 Troubleshooting

Common issues include:

- Missing groups → Check API permissions
- Redirect errors → Validate URI
- No DB entry → Review Prisma setup
- Token expired → Check sync
- Admin denied → Ensure group membership
- Enable Debug Logging:

```
NEXTAUTH_DEBUG=true
```

10 Features & Security Review

- Azure AD SSO
- Group-based role control
- Secure JWT session tokens
- Automatic user creation
- API route protection via NextAuth middleware
- Optional: Audit log, activity tracking, admin dashboard

11 Best Practices & Enhancements

- Email alerts for suspicious logins
- Audit trails for admin/security actions
- Periodic DB backups
- API rate limiting
- Session hardening
- Unit/integration testing (Jest/Playwright)

References

Graph Explorer | Try Microsoft Graph APIs - Microsoft Graph (2025). Available at: https://developer.microsoft.com/en-us/graph/graph-explorer (Accessed: 5 August 2025).

NextAuth.js (2025). Available at: https://next-auth.js.org (Accessed: 5 August 2025).

Overview – Vercel (no date). Available at: https://vercel.com/login?next=%2Fdocs (Accessed: 5 August 2025).

Prisma Documentation (2025). Available at: https://www.prisma.io/docs/ (Accessed: 5 August 2025).

rolyon (2025) *Microsoft Entra ID documentation - Microsoft Entra ID*. Available at: https://learn.microsoft.com/en-us/entra/identity/ (Accessed: 5 August 2025).