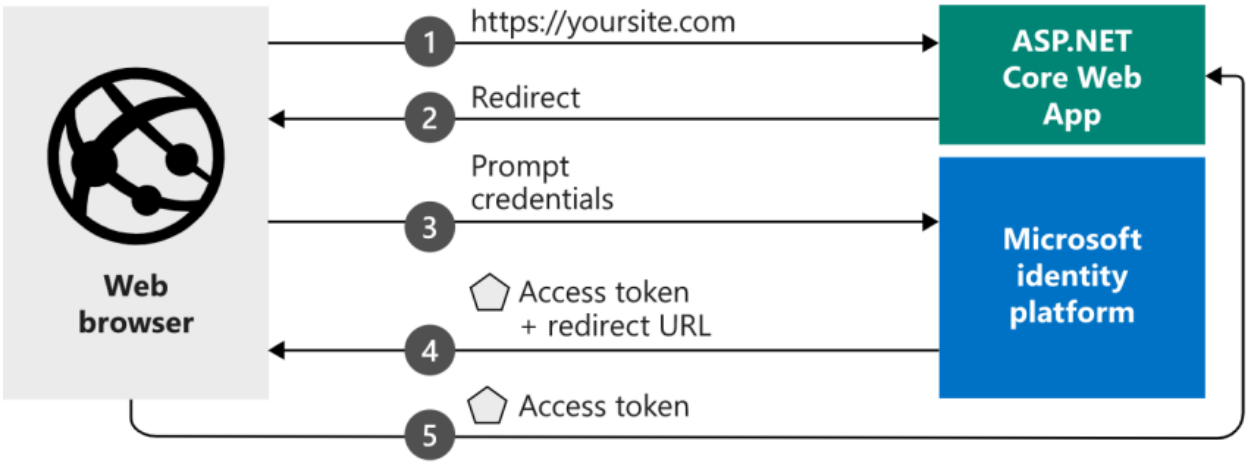
Summary:

We can implement Azure AD with web API and Angular application.

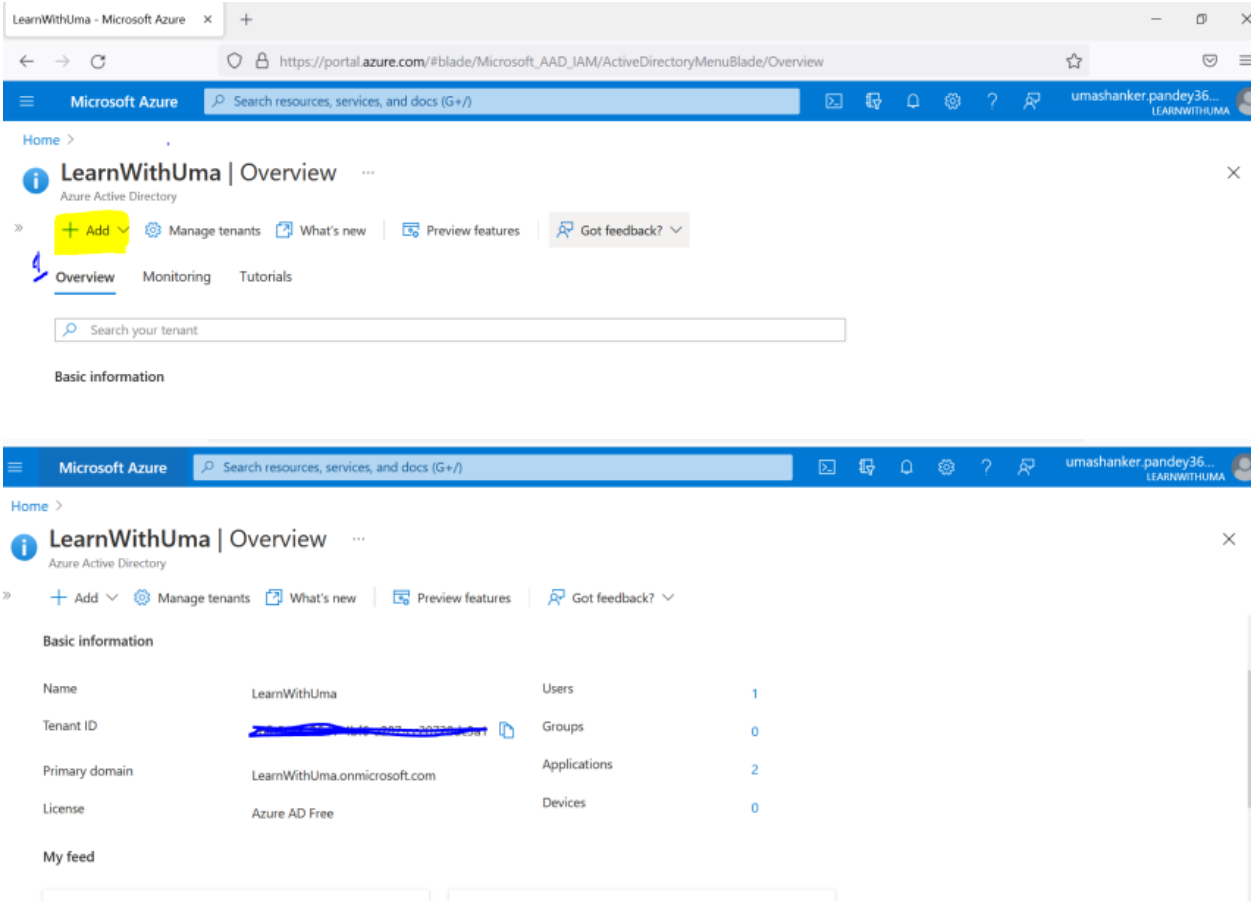


To connect azure ad with angular and asp.net core web API 5.0.

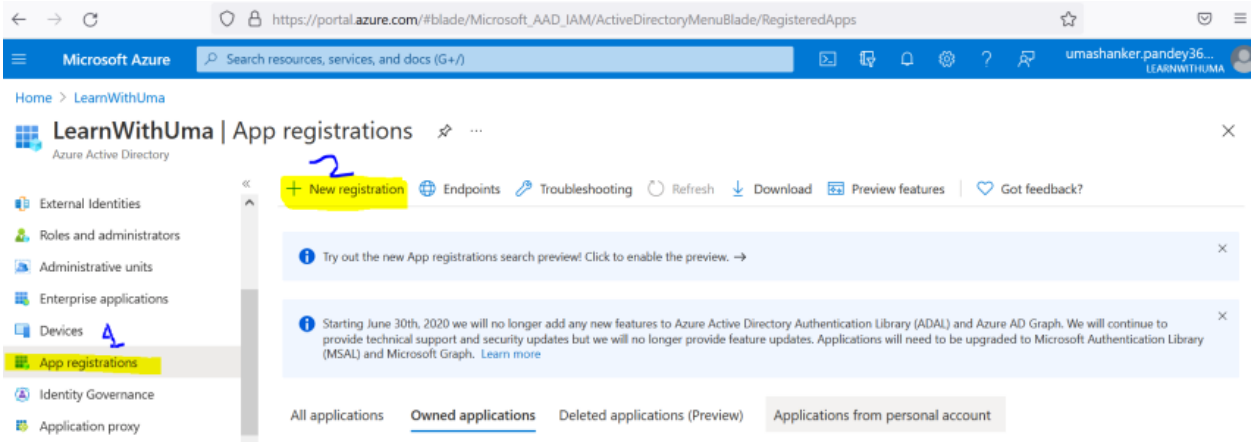
There are four-step processes to implement Azure AD in angular and asp.net core web API.

Step 1

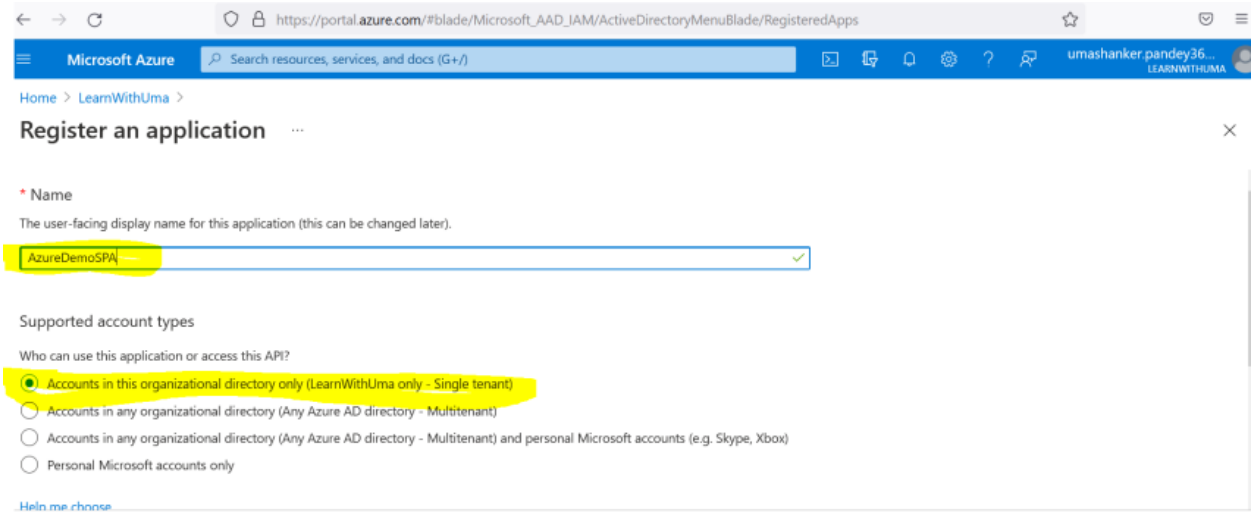
Create Azure AD Account and Register the SPA(Single Page Application) application in Azure AD App Registration blade.

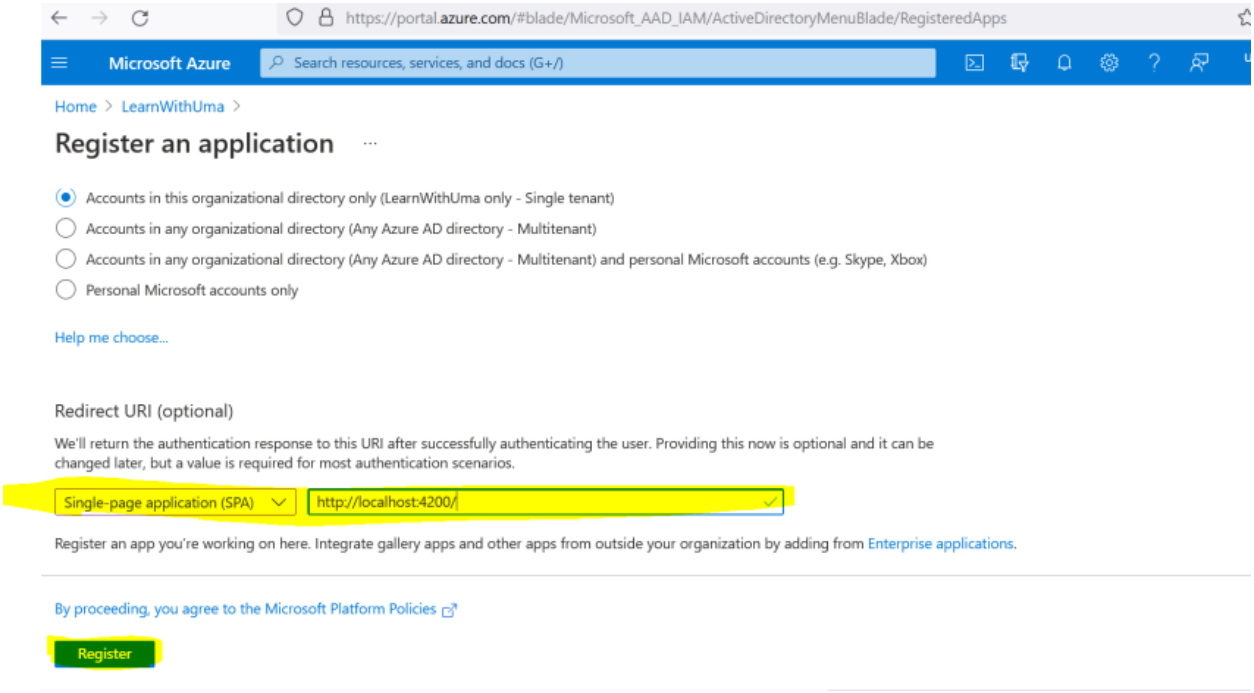


First, click on the App Registration button and then click on New Registration Button.

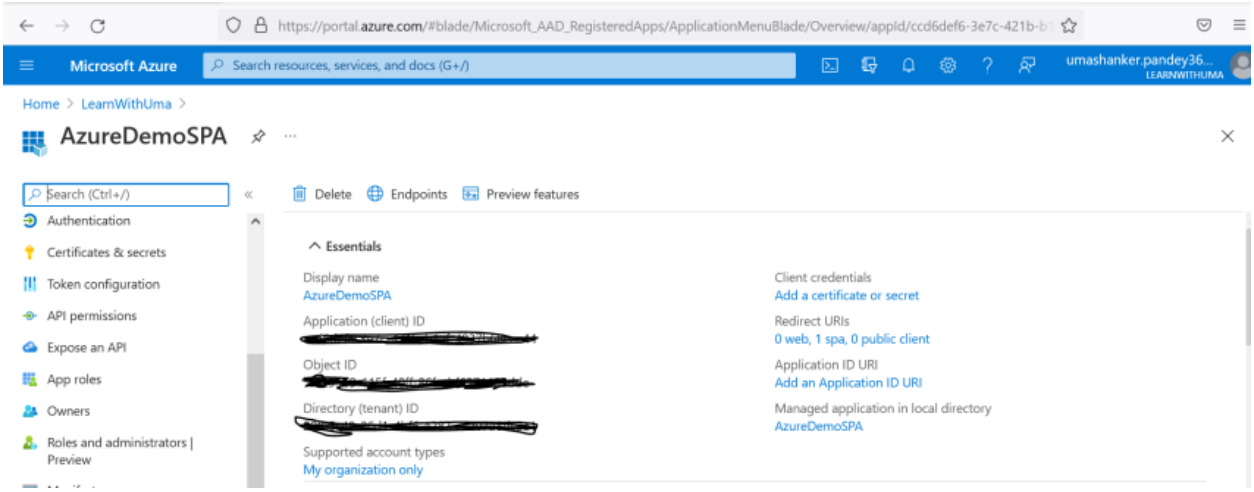


Fill in the Register Application Details.



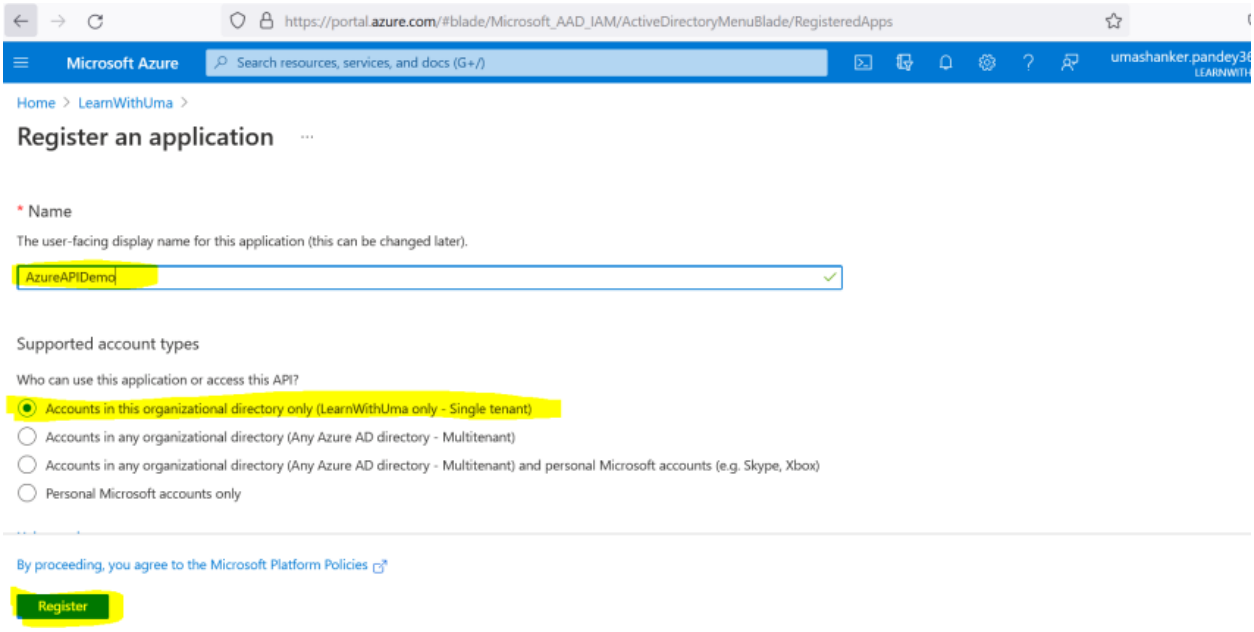


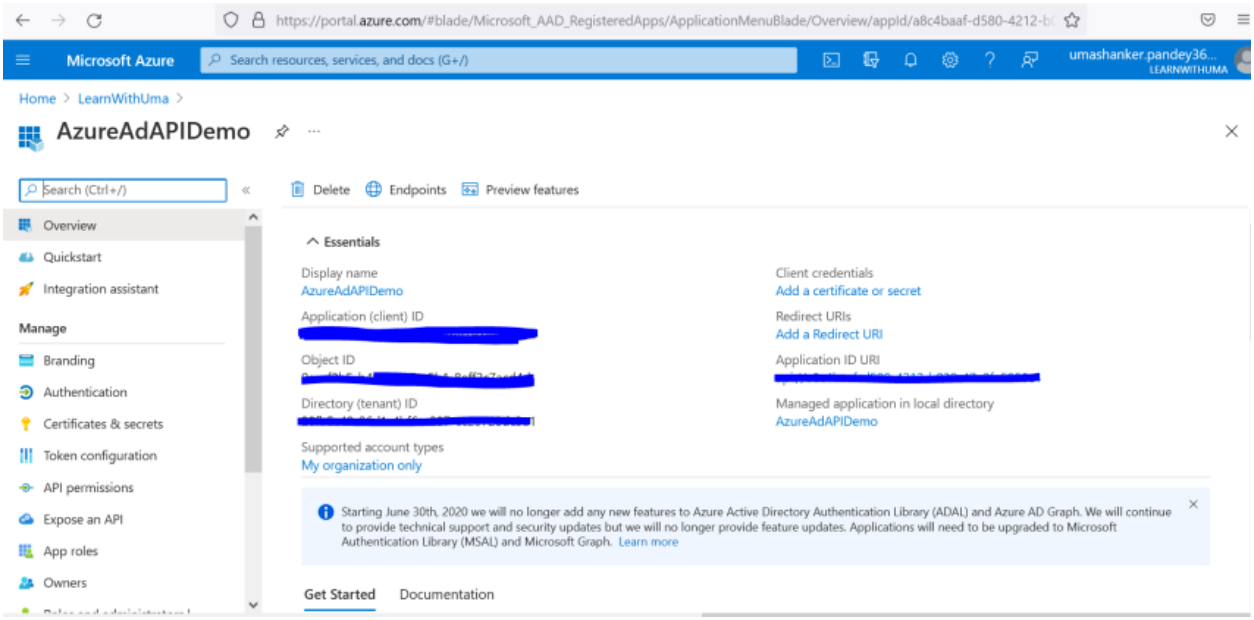
After clicking on the Register button. SPA application successfully gets registered and on the overview page, you get the register application details like Client ID, Tenant ID, etc.



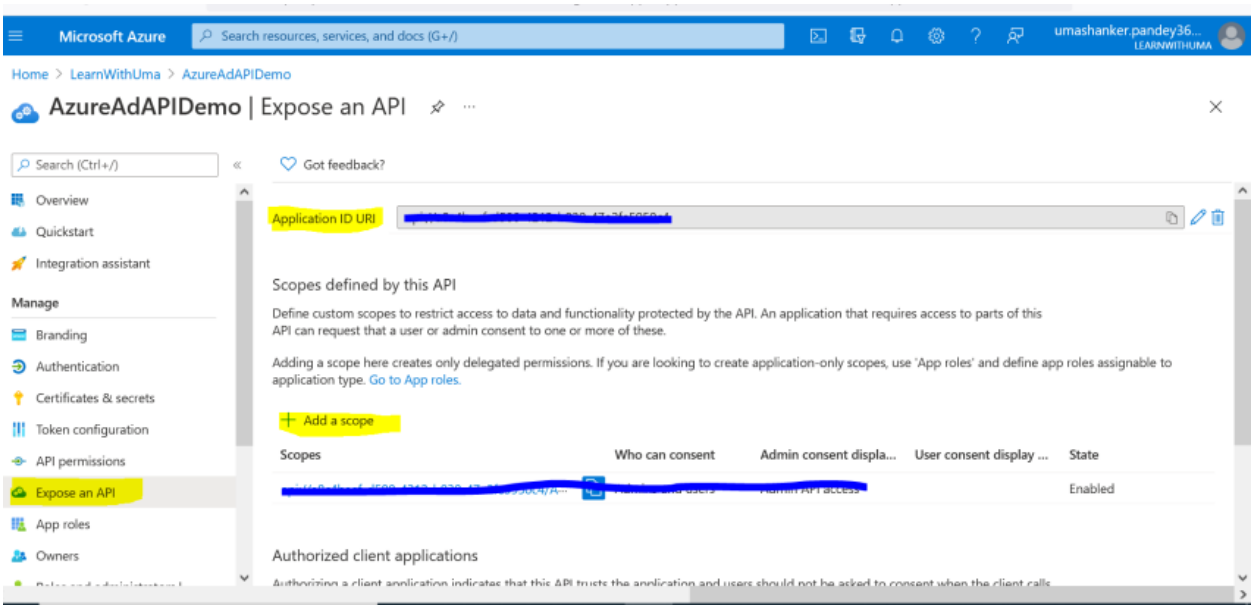
Step 2

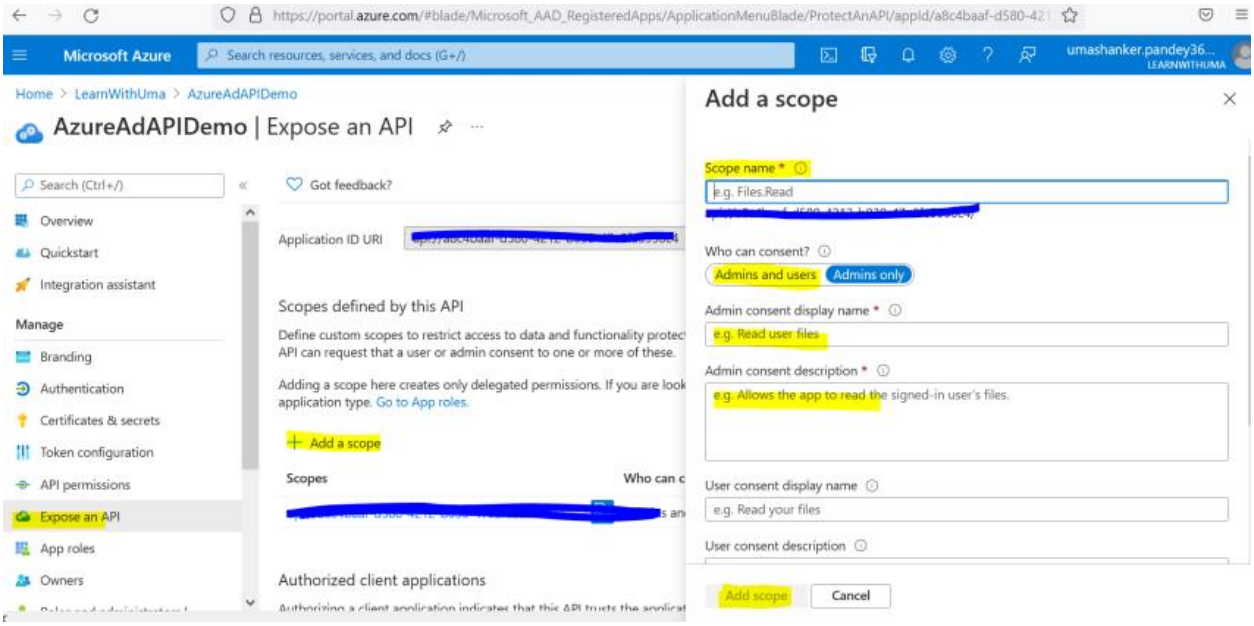
Register Web API applications in the same way. But in the API registration process, we do not need to provide the application redirect URL because our SPA Application is used to redirect the page.



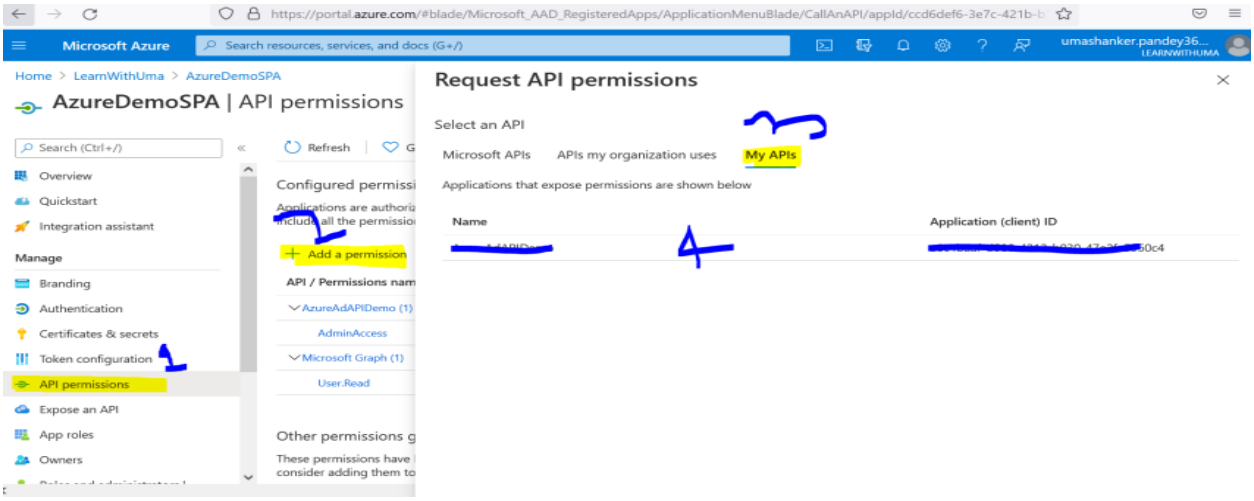


Then need to Expose An API,



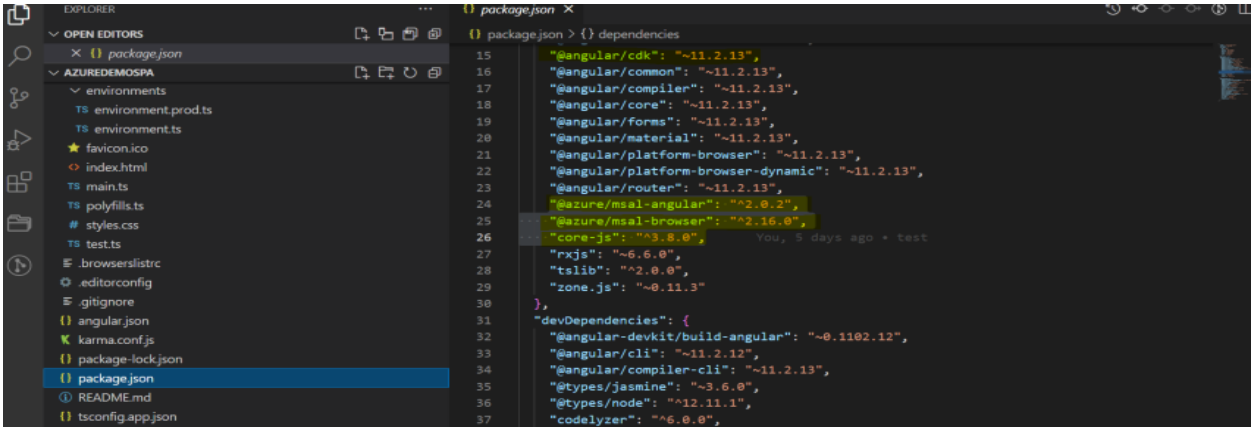


After successfully exposing the API and scope has been added you need to go into the SPA application and add the permission for this API. Follow the process and click as per the number in below pic.

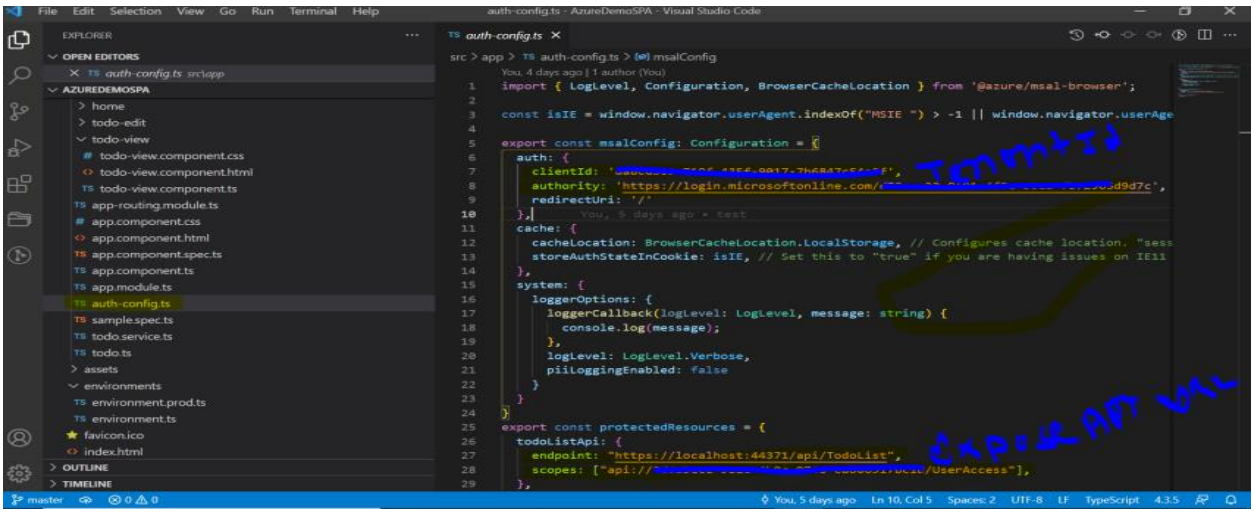


Step 3

Go to VS Code and Create New application in Angular, do the below changes. Add the below packages and do the npm install.



Add Auth-Config File and do the below settings as per the SPA application Registered.



Add below setting in App.Module.ts,

```
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { TodoEditComponent } from './todo-edit/todo-edit.component';
import { TodoViewComponent } from './todo-view/todo-view.component';
import { TodoService } from './todo.service';
import { HTTP_INTERCEPTORS, HttpClientModule } from
'@angular/common/http';
import { IPublicClientApplication, PublicClientApplication,
InteractionType } from '@azure/msal-browser';
import { MsalGuard, MsalInterceptor, MsalBroadcastService,
MsalInterceptorConfiguration, MsalModule, MsalService,
MSAL_GUARD_CONFIG, MSAL_INSTANCE, MSAL_INTERCEPTOR_CONFIG,
MsalGuardConfiguration, MsalRedirectComponent } from '@azure/msal-
angular';
import { msalConfig, loginRequest, protectedResources } from './auth-
config';
export function MSALInstanceFactory(): IPublicClientApplication {
    return new PublicClientApplication(msalConfig);
}
export function MSALInterceptorConfigFactory():
MsalInterceptorConfiguration {
    const protectedResourceMap = new Map < string,
        Array < string >> ();
    protectedResourceMap.set(protectedResources.todoListApi.endpoint,
protectedResources.todoListApi.scopes);
    return {
        interactionType: InteractionType.Redirect,
        protectedResourceMap
    };
}
export function MSALGuardConfigFactory(): MsalGuardConfiguration {
    return {
        interactionType: InteractionType.Redirect,
        authRequest: loginRequest
    };
}
@NgModule({
    declarations: [
        AppComponent,
        HomeComponent,
        TodoViewComponent,
        TodoEditComponent
    ],
```

```
    imports: [
        BrowserModule,
        BrowserAnimationsModule,
        AppRoutingModule,
        HttpClientModule,
        FormsModule,
        MsalModule
    ],
    providers: [{
            provide: HTTP_INTERCEPTORS,
            useClass: MsalInterceptor,
            multi: true
        }, {
            provide: MSAL_INSTANCE,
            useFactory: MSALInstanceFactory
        }, {
            provide: MSAL_GUARD_CONFIG,
            useFactory: MSALGuardConfigFactory
        }, {
            provide: MSAL_INTERCEPTOR_CONFIG,
            useFactory: MSALInterceptorConfigFactory
        },
        MsalService,
        MsalGuard,
        MsalBroadcastService,
        TodoService
    ],
    bootstrap: [AppComponent, MsalRedirectComponent]
})
export class AppModule {}
```

JavaScript

Copy

App.Component Settings,

```
import { Component, OnInit, Inject, OnDestroy } from '@angular/core';
import { MsalService, MsalBroadcastService, MSAL_GUARD_CONFIG,
MsalGuardConfiguration } from '@azure/msal-angular';
import { AuthenticationResult, InteractionStatus, InteractionType,
PopupRequest, RedirectRequest } from '@azure/msal-browser';
import { Subject } from 'rxjs';
import { filter, takeUntil } from 'rxjs/operators';
@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit, OnDestroy {
    title = 'Azure Ad Testing Demo';
    isIframe = false;
    loginDisplay = false;
    private readonly _destroying$ = new Subject < void > ();
    constructor(@Inject(MSAL_GUARD_CONFIG) private msalGuardConfig:
MsalGuardConfiguration, private authService: MsalService, private
msalBroadcastService: MsalBroadcastService) {}
    ngOnInit(): void {
        this.isIframe = window !== window.parent && !window.opener;
        /**
         * You can subscribe to MSAL events as shown below. For more
info,
         * visit: https://github.com/AzureAD/microsoft-authentication-
library-for-js/blob/dev/lib/msal-angular/docs/v2-docs/events.md
         */
        this.msalBroadcastService.inProgress$.pipe(filter((status:
InteractionStatus) => status === InteractionStatus.None),
takeUntil(this._destroying$)).subscribe(() => {
            this.setLoginDisplay();
        });
    }
    setLoginDisplay() {
        this.loginDisplay =
this.authService.instance.getAllAccounts().length > 0;
    }
    login() {
        if (this.msalGuardConfig.interactionType ===
InteractionType.Popup) {
            if (this.msalGuardConfig.authRequest) {
                this.authService.loginPopup({
                        ...this.msalGuardConfig.authRequest
                    }
                    as PopupRequest).subscribe((response:
AuthenticationResult) => {

this.authService.instance.setActiveAccount(response.account);
```

```
                });
            } else {
                this.authService.loginPopup().subscribe((response:
AuthenticationResult) => {

this.authService.instance.setActiveAccount(response.account);
                });
            }
        } else {
            if (this.msalGuardConfig.authRequest) {
                this.authService.loginRedirect({
                        ...this.msalGuardConfig.authRequest
                    }
                    as RedirectRequest);
            } else {
                this.authService.loginRedirect();
            }
        }
    }
    logout() {
        this.authService.logout();
    }
    // unsubscribe to events when component is destroyed
    ngOnDestroy(): void {
        this._destroying$.next(undefined);
        this._destroying$.complete();
    }
}
```

JavaScript

Copy

app.component.html settings,

```
<mat-toolbar color="primary">
  <a class="title" href="/">{{ title }}</a>
  <div class="toolbar-spacer"></div>
  <a mat-button [routerLink]="['todo-view']">TodoList</a>
  <button mat-raised-button *ngIf="!loginDisplay"
(click)="login()">Login</button>
  <button mat-raised-button color="accent" *ngIf="loginDisplay"
(click)="logout()">Logout</button>
</mat-toolbar>
<div class="container">
  <!--This is to avoid reload during acquireTokenSilent() because of
hidden iframe -->
  <router-outlet *ngIf="!isIframe"></router-outlet>
</div>
<footer *ngIf="loginDisplay">
  <mat-toolbar>
    <div class="footer-text"> How did we do? </div>
  </mat-toolbar>
</footer>
```

Markup
Copy

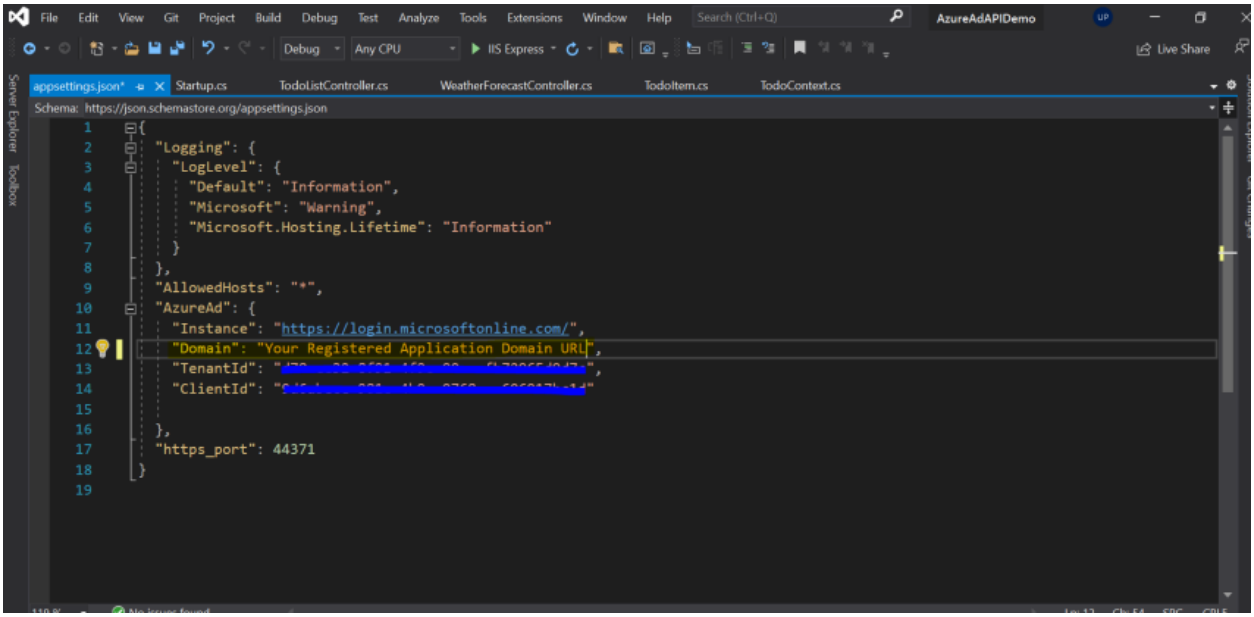
Index.html settings,

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>AzureDemoSPA</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.svg">
    <link
href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&displ
ay=swap" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
  </head>
  <body>
    <app-root></app-root>
    <app-redirect></app-redirect>
  </body>
</html>
```

Markup
Copy

Step 4 - Web API settings

In AppSettings.json file add your Registered API tenant and Client ID and Domain name information.



From Nuget Package manager add microsoft.identity.web package and do the changes in startup's file.

```
using AzureAdAPIDemo.Models;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Identity.Web;
using Microsoft.OpenApi.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace AzureAdAPIDemo {
    public class Startup {
        public Startup(IConfiguration configuration) {
            Configuration = configuration;
        }
        public IConfiguration Configuration {
            get;
        }
```

```
        // This method gets called by the runtime. Use this method to
add services to the container.
        public void ConfigureServices(IServiceCollection services) {
            // Setting configuration for protected web api

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).Add
MicrosoftIdentityWebApi(Configuration);
            // Creating policies that wraps the authorization
requirements
            services.AddAuthorization();
            services.AddDbContext < TodoContext > (opt =>
opt.UseInMemoryDatabase("TodoList"));
            services.AddControllers();
            // Allowing CORS for all domains and methods for the
purpose of the sample
            // In production, modify this with the actual domains you
want to allow
            services.AddCors(o => o.AddPolicy("default", builder => {

builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();
            }));
            services.AddSwaggerGen(c => {
                c.SwaggerDoc("v1", new OpenApiInfo {
                    Title = "AzureAdAPIDemo", Version = "v1"
                });
            });
        }
        // This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app,
IWebHostEnvironment env) {
            if (env.IsDevelopment()) {
                app.UseDeveloperExceptionPage();
                app.UseSwagger();
                app.UseSwaggerUI(c =>
c.SwaggerEndpoint("/swagger/v1/swagger.json", "AzureAdAPIDemo v1"));
            }
            app.UseCors("default");
            app.UseHttpsRedirection();
            app.UseRouting();
            app.UseAuthentication();
            app.UseAuthorization();
            app.UseEndpoints(endpoints => {
                endpoints.MapControllers();
            });
        }
    }
}
```

C#

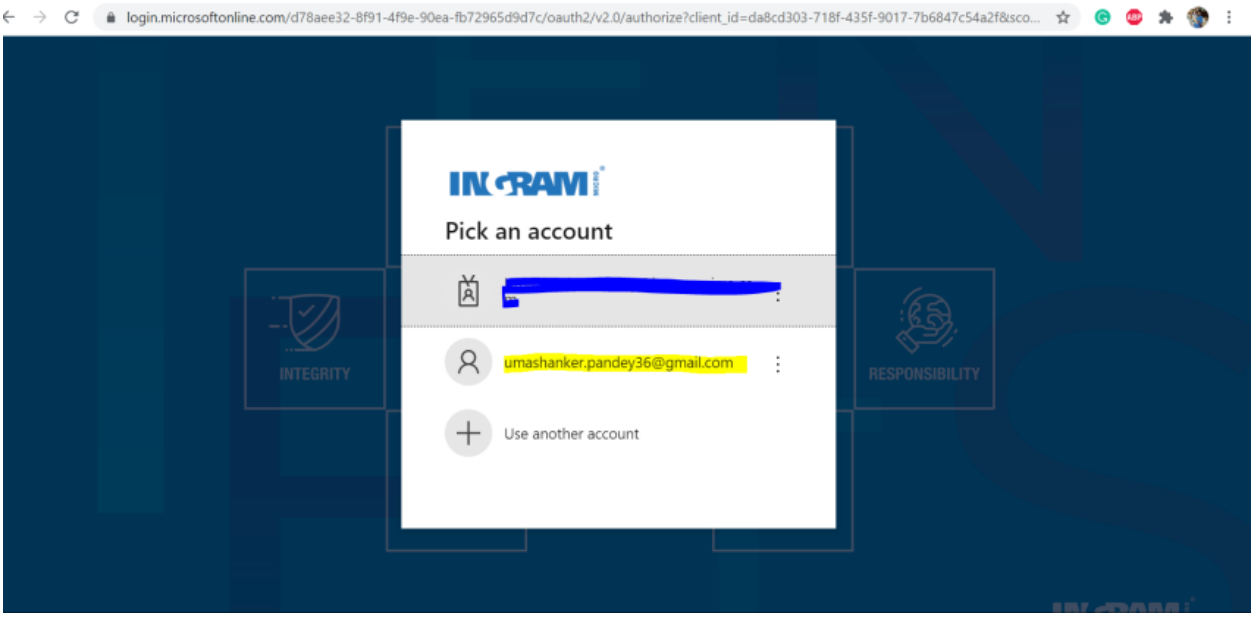
Copy

Then use [Authorize] on any controller.

```
{
    [Authorize]
    [Route("api/[controller]")]
    [ApiController]

    1 reference
    public class TodoListController : ControllerBase
    {
        // The Web API will only accept tokens 1) for users, and
        // 2) having the access_as_user scope for this API
        static readonly string[] scopeRequiredByApi = new string[] { "[redacted]" };

        private readonly TodoContext _context;

        0 references
        public TodoListController(TodoContext context)
        {
            context = context;
```

No issues found

After clicking the login button,



Select the account and log in successfully.