**Angular**

**Base Service File (base-http.service.ts)**

constructor(private httpClient: HttpClient, protected apiConfig: ConfigDataProvider, ) { }

  //baseUrl: string = 'http://192.168.254.35:8087/';

  baseUrl: string = 'http://localhost:58853/';

  /\*\*

   \* Function used for HTTP get

   \* @param relativeUrl API endpoint

   \* @param options Optional parameters

   \*/

  public get<T, OT = any>(relativeUrl: string, options?: OT) {

    if (options) {

      return this.httpClient.get<T>(`${this.baseUrl}${relativeUrl}`, options );

    } else {

      return this.httpClient.get<T>(`${this.baseUrl}${relativeUrl}` );

    }

  }

  /\*\*

   \* Function used for HTTP get post

   \* @param relativeUrl API endpoint

   \* @param data Data object

   \* @param options Optional parameters

   \*/

  public post<T, OT = any>(relativeUrl: string, data: T, options?: OT) {

    if (options ) {

      return this.httpClient.post<T>(`${this.baseUrl}${relativeUrl}`, data, options);

    } else {

      return this.httpClient.post<T>(`${this.baseUrl}${relativeUrl}`, data);

    }

  }

  /\*\*

   \* Function used for HTTP put

   \* @param relativeUrl API endpoint

   \* @param data Data object

   \* @param options Optional parameters

   \*/

  public put<T, OT = any>(relativeUrl: string, data?: T, options?: OT) {

    if (options ) {

      return this.httpClient.put<T>(`${this.baseUrl}${relativeUrl}`, data, options);

    } else {

      return this.httpClient.put<T>(`${this.baseUrl}${relativeUrl}`, data);

    }

  }

  /\*\*

   \* Function used for HTTP delete

   \* @param relativeUrl: API endpoint

   \* @param options: Optional parameters

   \*/

  public delete<T, OT = any>(relativeUrl: string, options?: OT) {

    if (options) {

      return this.httpClient.delete<T>(`${this.baseUrl}${relativeUrl}`, options );

    } else {

      return this.httpClient.delete<T>(`${this.baseUrl}${relativeUrl}` );

    }

  }

**Login Screen (**login.component.ts)

 /\*\*

  \* Method to called login user api

  \* @param respData

  \*/

  callLoginUserApi(respData: any) {

    this.isDataLoading=true;

    this.authService.loginExistingUser(respData)

      .pipe(takeUntil(this.onDestroy$))

      .subscribe({

        next: (retData: any) => {

          if (retData.status) {

            this.toastService.successMessage(Messages.RegisterUserSuccess);

            this.router.navigate(['home']);

          } else {

            this.toastService.errorMessage(retData.message);

          }

          this.isDataLoading = false;

        },

        error: (err: any) => {

          console.log(err);

          this.isDataLoading = false;

        },

        complete: () => {

          console.log('complete');

          this.isDataLoading = false;

        }

      });

  }

}

**Registration Screen (**registration.component.ts)

 /\*\*

   \* Method to called register user api

   \* @param respData

   \*/

  callRegisterUserApi(respData: any) {

    this.isDataLoading=true;

    this.authService.registerNewUser(respData)

      .pipe(takeUntil(this.onDestroy$))

      .subscribe({

        next: (retData: any) => {

          this.isDataLoading = false;

          if (retData.status) {

            this.toastService.successMessage(Messages.RegisterUserSuccess);

            this.router.navigate(['login']);

          } else {

            this.toastService.errorMessage(retData.message);

          }

        },

        error: (err: any) => {

          console.log(err);

          this.isDataLoading = false;

        },

        complete: () => {

          console.log('complete');

          this.isDataLoading = false;

        }

      });

  }

}