**Technical Documentation of ApneSaathi Backend Portal**

Contents

[**1. Technologies Used** 2](#_Toc52382717)

[**3. Login Component** 3](#_Toc52382718)

[**4. protect.gurad.ts** 3](#_Toc52382719)

[**5. Header Component** 4](#_Toc52382720)

[**6. Menu Component** 4](#_Toc52382721)

[**7. Modules** 5](#_Toc52382722)

[**i) Volunteers** 5](#_Toc52382723)

[**ii) Senior-Citizens** 6](#_Toc52382724)

[**8. Global Popup** 7](#_Toc52382725)

[**9. notification-message** 8](#_Toc52382726)

[**10. material module** 8](#_Toc52382727)

[**11. environments variables** 9](#_Toc52382728)

## **1. Technologies Used**

i) Angular 10

ii) HTML

iii) CSS

iv) Javascript (ES6)

**2. Folder structure of Project**

**header**

**menu**

**login**

**global-dialog**

**notification-message Components**

**material Modules**

**volunteers Services**

**add-volunteers Class Files**

**assign-volunteers**

**de-board-volunteer**

**import-log**

**transfer-volunteer**

**volunteer-detail-view**

**volunteers-list**

**volunteers.module.ts**

**volunteers-routing.module.ts**

**senior-citizens**

**assign-senior-citizens**

**senior-citizens-list**

**senior-citizens.module.ts**

**senior-citizens-routing.module.ts**

**Services**

**api-info.service.ts**

**api.interceptor.ts**

**shared.service.ts**

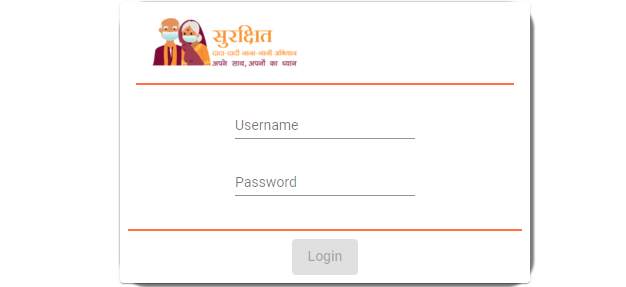
**location.service.ts**

**protect.guard.ts**

The above folder structure specifies the folders and files that are created for the project apart from the default files and folders that comes along with the Angular project setup.

## **3. Login Component**

Admin can login into the application using username and password.



**Figure: Login**

All the login related code can be found in login component of angular application.

## **4. protect.gurad.ts**

Every module and respective components of ApneSaathi Angular application can be accessible only to authorized/login users only.

Since we are using angular application we have used **canActivate**  interface which is located in **protect.guard.ts** file to protect the Routing of application.

To protect a route we need to add canActivate property to the routing object. Find the below code for adding canActivate property to the routing object.

*const routes: Routes = [*

*{*

*path: '', pathMatch: 'full', redirectTo: 'volunteers'*

*},*

*{*

*path: 'login', component: LoginComponent*

*},*

*{*

*path: 'volunteers',*

*loadChildren: () => import('./volunteers/volunteers.module').then(m => m.VolunteersModule),*

***canActivate: [ProtectGuard]***

*},*

*{ path: 'seniorCitizens',*

*loadChildren: () => import('./senior-citizens/senior-citizens.module').then(m => m.SeniorCitizensModule),*

***canActivate: [ProtectGuard]***

*}*

*];*

**canActivate** is the interface that is defined inside protect.guard.ts file. Below is the definition

*canActivate(*

*next: ActivatedRouteSnapshot,*

*state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {*

*// logged in so return true*

*if(this.sharedService.validateLogin()){*

*this.sharedService.removeAppLocalStorage("returnUrl");*

*return true;*

*}*

*else{*

*// not logged in so redirect to login page with the return url*

*this.sharedService.setAppLocalstorage("returnUrl",state.url)*

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

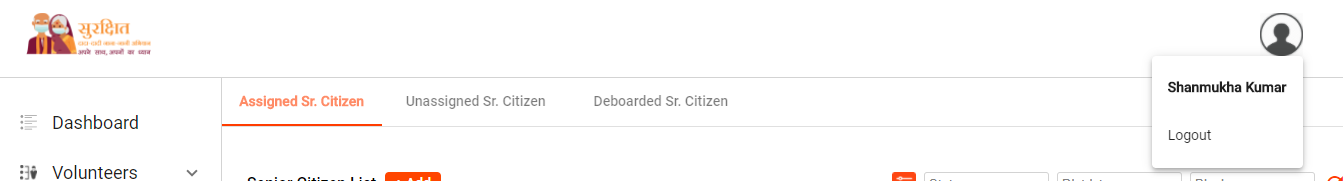
*return false;*

*}*

*}*

## **5. Header Component**

Header part consists of Project logo and User related details of the application. We can find User logo on the right end side of the Header

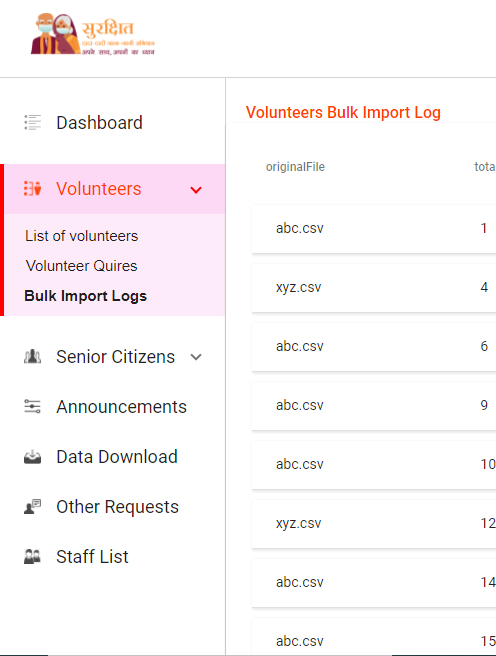


**Figure 1: Header**

All the code related with header part is located in the header component of the angular application. Also, the logout related code also located in the header.component.ts file

## **6. Menu Component**

All the Menu related code can be found in menu component of angular application.



**Figure: Menu**

## **7. Modules**

In this ApneSaathi application we do have mainly two modules one is Volunteers and other one is Senior Citizens.

### **i) Volunteers**

Volunteers module is the heart of the application. Below are few main components under the volunteers module

1. **volunteers-list**

All the code related with Active volunteers list , Deboarded volunteers list can be found here inside volunteers-list component. Admin can perform actions such as Assigning senior citizens to specific volunteer, transfer of location of particular volunteer and Deboard volunteer from the Active volunteers list page.

1. **add-volunteers**

Code related with bulk uploading of volunteers through csv can be found in this add-volunteers component. Since the add volunteer feature needs to be shown in the global popup component which is not part of volunteers module but app.module. we export add-volunteers component from the volunteers.module.ts file and import in app.module.ts file.

Please find the export related code in the assign-volunteers component description which is point no 4.

Below is the code for importing volunteers module(app.module.ts):

*import { VolunteersModule } from './volunteers/volunteers.module';*

*@NgModule({*

*declarations: [...],*

*imports: [ VolunteersModule ]*

*})*

1. **Import-logs**

Code related with displaying bulk upload track of volunteers can be found here. We can find logic to download the error csv file which gives the records that are not get inserted due to some reasons.

1. **assign-volunteers**

Code related with assigning volunteers to the senior citizens in the popup can be found here in this assign-volunteers component. Since the assigning of volunteers or distributing of volunteers among senior citizens gets displayed in global popup which is not part of the volunteers module, we export the assign-volunteers component from the volunteers.module.ts file. Below is the code

*@NgModule({*

*declarations: [VolunteersComponent, VolunteersListComponent, VolunteerDetailViewComponent, AddVolunteersComponent, TransferVolunteerComponent, DeBoardVolunteerComponent,* ***AssignVolunteersComponent****, ImportLogComponent],*

*imports: [*

*CommonModule,*

*CarouselModule,*

*VolunteersRoutingModule,*

*MaterialModule,*

*ReactiveFormsModule,*

*NgxPaginationModule,*

*FormsModule*

*],*

*exports:[AddVolunteersComponent,TransferVolunteerComponent,DeBoardVolunteerComponent,****AssignVolunteersComponent****]*

*})*

1. **de-board-volunteer**

All the code related with deboarding volunteer can be located in this de-board-volunteer component. Since it is shown on the global popup the component is exported from volunteers.module.ts file

1. **volunteer-detail-view**

The code related with detail view of volunteer which includes person info, call summary, rating and assigned senior citizens can be found here in this volunteer-detail-view component. User/Admin can add new senior citizens to the volunteer and also Admin can deboard volunteer from this detail view page.

### **ii) Senior-Citizens**

Unlike volunteers module senior-citizens module is having less features and components. Below are the details of components

1. **senior-citizens-list**

The code related with list of Active, Un assigned and deboarded senior citizens can be found in this component.

Admin can distribute unassigned senior citizens by selecting citizens from the Unassigned list.

Admin can add more senior citizens thorough bulk upload of senior citizens from the Active senior citizens list.

1. **assign-senior-citizens**

When we click on assign senior citizens from the volunteers list or volunteer detail view. A global popup will get opened with the list of unassigned senior citizens from the district and state of selected volunteer. The code relate with showing list and assigning senior citizens to volunteer can be found in this assign-senior-citizens component. Since we are binding the component in the popup we export the assign-senior-citizens component from the senior-citizens module to the app module

## **8. Global Popup**

Global popup is the component which is the skeleton of all the popups of application. Code related Header and title parts of popup, size and design of popup can be found here. The body part of the Popup gets bind based on the action that we are going to display in popup. Below is the code of dynamic integration of component in the goabl-popup.component.html

*<div mat-dialog-title [ngClass]="{'heading-subs-right-container':data.headingRightContent}">*

*<h3 >*

*{{data.heading}}*

*<div class="subscript-text gray-text"><sub \*ngIf="data.headingSubscript"><i>{{data.headingSubscript}}</i></sub></div>*

*</h3>*

*<div class="heading-right-content">*

*{{data.headingRightContent}}*

*</div>*

*</div>*

*<div>*

*<ng-template [ngIf]="data.feature=='assignCitizensSingleVolunteer'">*

*<app-assign-senior-citizens [volunteerObj]="data.volunteerObj"></app-assign-senior-citizens>*

*</ng-template>*

*<ng-template [ngIf]="data.feature=='addVolunteer'">*

*<app-add-volunteers></app-add-volunteers>*

*</ng-template>*

*<ng-template [ngIf]="data.feature=='deboardingVolunteer'">*

*<app-de-board-volunteer [volunteerObj]="data.volunteerObj"></app-de-board-volunteer>*

*</ng-template>*

*<ng-template [ngIf]="data.feature=='assignSrCitizensEqually' || data.feature=='distributeSrCitizensEqually'">*

*<app-assign-volunteers [volunteerObj]="data.volunteerObj ? data.volunteerObj:{}" [citizensObj]="data.citizensObj ? data.citizensObj: {}" [inputObj]="data.inputObj ? data.inputObj:{}"></app-assign-volunteers>*

*</ng-template>*

*</div>*

## **9. notification-message**

Code related with the design of snackbar messages of angular are located in this notification-message component. Below is the code.

*<div class="notification-container">*

*<div class="success-icon" \*ngIf="data.success">*

*<div class="tick1"></div>*

*<div class="tick2"></div>*

*</div>*

*<div class="notification-message">{{data.message}}</div>*

*</div>*

Message to display gets passed to the notification-message component which executing code of showing snackbar message. Below is sample function to display snackbar message.

*abc(){*

*this.showNotification({message:'This is snackbar message',success:true});*

*}*

*showNotification(notificationData,duration=5000){*

*this.snackBar.openFromComponent(NotificationMessageComponent,{*

*data:notificationData,*

*duration:duration,*

*panelClass: "notification-snackbar"*

*});*

*}*

## **10. material module**

Material module consists of common class file which loads/imports all the Angular material related modules. This material.module file can be imported in any of the modules inside application so that the module which is importing material.module file does not need importing Angular material related modules sepetately.

Sample code inside material.model.ts file

*import { NgModule } from '@angular/core';*

*import {MatListModule} from '@angular/material/list';*

*import {MatMenuModule} from '@angular/material/menu';*

*import {MatIconModule} from '@angular/material/icon';*

*const matVariables=[*

*MatListModule,*

*MatMenuModule,*

*MatIconModule,*

*]*

*@NgModule({*

*imports: [matVariables],*

*exports:[matVariables]*

*})*

*export class MaterialModule { }*

Code to import material module file in volunteer.module.ts

*import {MaterialModule} from '../material/material.module';*

*@NgModule({*

*declarations: [...],*

*imports: [ MaterialModule]*

*})*

## **11. environments variables**

We get environments folder by default when we setup our angular application through Angular CLI. However we are using locahost for development purpose and Apace server for deployment. We should take the path of assets of application dynamically based on environment.

So, environment.ts file and environment.prod.ts file has been updated.

**environment.ts**

*export const environment = {*

*production: false,*

*base\_url: 'http://localhost:4200/'*

*};*

**environment.prod.ts**

*export const environment = {*

*production: true,*

*base\_url: 'http://15.207.42.209:8080/apneSaathiAngular/'*

*};*