To navigate between 2 tabs on condition basis.

HTML

<a \*ngIf="screenView" (click)="tableViewScreen()">Back to all patient claims</a>

<ng-container \*ngFor="let panel of panelConfig">

    <div [class]="panel.direction === 'right' ? 'pd-right-col': 'pd-left-col'">

        <ng-container \*ngFor="let section of panel.sections">

            <div [class]="section.class">

                <ng-container [ngTemplateOutlet]="section.heading === 'claims' ? table : grid"

                    [ngTemplateOutletContext]="{$implicit: section}">

                </ng-container>

            </div>

        </ng-container>

    </div>

</ng-container>

<!-- To display table and grid based on screen -->

<ng-template #table let-section>

    <z-simple-table [patientClaimDetails]="section.config" (patientDetail)="getPatientDetails($event)"></z-simple-table>

</ng-template>

<ng-template #grid let-section>

    <div>

        <h5 [innerHTML]="section.heading"></h5>

        <z-dynamic-form #dynamicForm

            [fieldsConfig]="section.config"></z-dynamic-form>

    </div>

</ng-template>

TS:

 @Input() patientClaimDetails: any;

  panelConfig: any;

  screenView: boolean;

  constructor(private detailService: DetailViewServiceService) { }

  ngOnChanges() {

    this.fieldConfigModefication(this.patientClaimDetails);

  }

  getPatientDetails(event: any) {

    const payload = {

      viewName: "PatientClaimDetails",

      patientId: 14187528,

      claimId: event.claimId,

      includeNotesAndFiles: false

    }

    this.detailService.getPatientClaimDetails(payload).subscribe((res: any) => {

      if (res) {

        this.panelConfig = [];

        this.fieldConfigModefication(res);

        this.screenView = true;

      }

    })

  }

  fieldConfigModefication(list: any) {

    let leftPanelSections: Array<object> = [];

    let rightPanelSections: Array<object> = [];

    list.sections.forEach((element: any) => {

      let fieldsConfiguration: Array<object> = [];

      let data

      if (element.section == 'claims') {

        this.screenView = false;

        data = {

          "heading": 'claims',

          "config": list

        };

      }

      else {

        element.fields.forEach((config: any) => {

          fieldsConfiguration.push({

            field: config.field,

            size: 1,

            type: config.type,

            value: list?.data[element?.section] ? list?.data[element?.section][config?.field] : null,

            label: config.displayName,

            options: '',

          });

        });

        data = {

          "heading": element.sectionName,

          "config": fieldsConfiguration

        };

      }

      element.section == 'facilityDetails' || element.section == 'teamInformation' ? rightPanelSections.push(data) : leftPanelSections.push(data);

    });

    this.panelConfig = [{

      sections: leftPanelSections,

      direction: "left"

    },

    {

      sections: rightPanelSections,

      direction: "right"

    }];

  }

  tableViewScreen(){

    this.fieldConfigModefication(this.patientClaimDetails);

  }

}

**To dynamically create a search builder with configuration received from API:**

HTML

<div \*ngIf="searchBuilderConf.length" class="search-builder flex align-items-center gap-2 p-1 pl-2 pr-2 flex-wrap">

    <z-dynamic-form [fieldsConfig]="searchBuilderConf" (valueChangesEve)="subDropDownChanges()"></z-dynamic-form>

    <z-button class="search-filter" (handleClick)="searchClick()">Search</z-button>

</div>

TS:

@ViewChild(DynamicFormComponent) dynamicForm: DynamicFormComponent;

  @Input() pageLocation: string;

  @Input() searchBuilderConfiguration: any;

  @Output() searchResult: any = new EventEmitter();

  searchBuilderConf: Array<FieldConfig> = [];

  constructor(private utils: UtilityService) {

  }

  ngOnChanges() {

    this.searchBuilderConf = [];

    let pageName = this.searchBuilderConfiguration?.pageName;

    let configuration = this.searchBuilderConfiguration?.configuration;

    configuration?.search?.forEach((data: any) => {

      let dropDownVal: any = [];

      let defaultValue: any = '';

      // TODO: Text to be changed to searchable dropdown key

      if (data.type === fieldTypes.dropdown || data.type === fieldTypes.multi || data.type === fieldTypes.searchInField) {

        dropDownVal = this.bindDropdown(data.type === fieldTypes.searchInField ? fieldTypes.searchInField : data.field, configuration, defaultValue?.value || null);

        if (data.field.toLowerCase() === searchBuilderItems.claimsubstatus) {

          defaultValue = this.searchBuilderConf.find((x: any) => x.field === searchBuilderItems.claimStatus);

        } else if (data.field === searchBuilderItems.claimStatus) {

          defaultValue = dropDownVal.find((x: any) => pageName.includes(x.value));

        }

      }

      this.searchBuilderConf.push({

        field: data.field,

        size: 1,

        type: data.type,

        value: defaultValue?.key || '',

        label: data.displayName,

        options: dropDownVal

      });

    });

  }

  bindDropdown(field: any, configuration: any, value: any) {

    let rowData;

    if (field === 'claimSubStatus') {

      rowData = {

        claimStatus: value

      };

    }

    return this.utils.bindDropdowns(configuration, field, rowData);

  }

  searchBuilerConfiguration(searchBuilderVal: any, selectedMenuSteps: Array<string>) {

    return selectedMenuSteps.reduce(

      (step, index: any) => (step && step[index] !== null && step[index] != undefined ? step[index] : null),

      searchBuilderVal

    );

  }

  subDropDownChanges() {

    this.dynamicForm.form.controls[searchBuilderItems.claimStatus].valueChanges.subscribe(val => {

      this.dynamicForm.setValue('claimSubStatus', null);

      let configuration = this.searchBuilderConfiguration?.configuration;

      this.searchBuilderConf.map((data: any) => {

        if (data.field === 'claimSubStatus') {

          data.options = this.bindDropdown(data.field, configuration, val);

        }

      });

    });

  }

  searchClick() {

    if (this.dynamicForm.form.valid) {

      this.searchResult.emit(this.dynamicForm.form);

    } else {

      (<any>Object).values(this.dynamicForm.form.controls).forEach((control: any) => {

        control.markAsTouched();

        if (control.invalid) {

          control.markAsDirty();

        }

      });

    }

  }

**To show the data in tabular form**

HTML:

<p-table dataKey="id" [columns]="columnsHeader" [value]="tableData" class="facility-info-grid">

    <ng-template pTemplate="header" let-columns>

        <tr>

            <ng-container \*ngFor="let col of columns">

                <th \*ngIf="col.type !=='details' && col.type !=='expand'" [pSortableColumn]="col.sortable ? col.field: null">{{col.displayName}} <p-sortIcon \*ngIf="col.sortable" [field]="col.field"></p-sortIcon></th>

                <th \*ngIf="col.type =='details'" class="float-right">{{col.displayName}}</th>

                <th \*ngIf="col.type =='expand'" class="float-left">{{col.displayName}}</th>

            </ng-container>

        </tr>

    </ng-template>

    <ng-template pTemplate="body" let-rowData let-columns="columns" let-expanded="expanded">

        <tr>

            <ng-container \*ngFor="let col of columns">

                <td \*ngIf="col.type !=='details' && col.type !=='expand'">

                    {{rowData[col.field]}}

                </td>

                <td \*ngIf="col.type ==='details'" class="text-center float-right">

                    <i \*ngIf="!rowData?.children" class="pi pi-exclamation-circle" (click)="showPatientDetails($event, rowData)"></i>

                </td>

                <td \*ngIf="col.type ==='expand'" class="float-left">

                    <button \*ngIf="col.type === 'expand' && rowData?.children" type="button" pButton pRipple

                        [pRowToggler]="rowData" class="p-button-text p-button-rounded p-button-plain"

                        [icon]="expanded ? 'pi pi-minus-circle' : 'pi pi-plus-circle'"></button>

                </td>

            </ng-container>

        </tr>

    </ng-template>

    <ng-template pTemplate="rowexpansion" let-index="rowIndex" let-rowData let-columns="columns">

        <ng-container \*ngFor="let child of rowData.children">

            <tr class="expended-row">

                <ng-container \*ngFor="let col of columns">

                    <td \*ngIf="col.type !=='details' && col.type !=='expand'" [class]="{'tree-icon' : col.type === 'expand' && rowData.children}">

                        {{child[col.field]}}

                    </td>

                    <td \*ngIf="col.type ==='expand'" class="float-left"></td>

                    <td  class="text-center float-right">

                        <i \*ngIf="col.type ==='details'" class="pi pi-exclamation-circle" (click)="showPatientDetails($event, child)"></i>

                    </td>

                </ng-container>

            </tr>

        </ng-container>

    </ng-template>

</p-table>

TS:

@Input() patientClaimDetails: any;

  columnsHeader: any = [];

  tableData: any = [];

  @Output() patientDetail = new EventEmitter<any>();

  constructor() { }

  ngOnChanges() {

    this.columnsHeader = this.patientClaimDetails.sections[0].fields;

    this.tableData = this.patientClaimDetails.data?.claims;

  }

  showPatientDetails(event: any, rowData: any){

    this.patientDetail.emit(rowData);

  }

**To dynamically create navigation**

HTML:

<div class="flex align-items-center nav-wrapper justify-content-between gap-3 pl-2 pr-2 header-bar">

    <div class="navigation flex align-items-center gap-3">

        <img class="site-logo" [src]="logoUrl" alt="{{logoAlt}}">

        <p-menubar #tableMenu [model]="navMenu" appendTo="body">

        </p-menubar>

    </div>

    <div class="nav-icon flex gap-3 align-items-center">

        <i class="pi pi-question-circle cursor-hand"></i>

        <i class="pi pi-cog cursor-hand"></i>

        <i class="pi pi-user cursor-hand" (click)="onPersonIconClick($event)"></i>

        <p-menu #menu [popup]="true" styleClass="profile-menu" [model]="userDetail"></p-menu>

    </div>

</div>

TS:

@ViewChild('menu') menu: any;

  navMenu: MenuItem[] = [];

  logoAlt: string = 'Site Logo';

  logoUrl: string = '';

  isInitialised: boolean = false;

  userDetail: any;

  toggle: boolean = true;

  userImagePath: string = '';

  constructor(

    private sessionStorageService: SessionStorageService,

    private eventService: EventService,

    private sessionStorage: SessionStorageService,

    private gloginService: GloginService) { }

  ngOnInit(): void {

    let applicationDetail: AppDetail = this.sessionStorageService.getData(StorageKeys.appDetail);

    this.logoUrl = applicationDetail ? applicationDetail['logoUrl'] : '';

    const userDetail = this.sessionStorage.getData(StorageKeys.userDetail);

    this.userImagePath = userDetail?.image || '../../assets/images/user.svg';

    const headerData = this.sessionStorage.getData(StorageKeys.headerDetail);

    if (headerData?.menu) {

      this.navMenuConfiguration(headerData.menu, this.navMenu);

    }

    this.userDetail = [

      {

        label: '<div class="user-wrapper"><img class="user-image" src="' + this.userImagePath + '"> </img> <span class="user-name">' + `${userDetail?.name || 'Zealie User'}` + '</span><span class="user-role">' + `${headerData?.role || ''}` + '</span></div>',

        escape: false,

        items: [

          {

            separator: true,

          },

          {

            label: "Logout",

            icon: 'pi pi-fw pi-power-off',

            command: () => {

              this.logOut();

            },

          }

        ]

      }

    ];

  }

  navMenuConfiguration(navItem: any, configurationMenu: MenuItem[], itemLocation: string = "") {

    Object.keys(navItem).forEach((navElement: any, index) => {

      let menuItem: MenuItem = {

        label: navElement

      };

      let subMenuItem = navItem[navElement] || {};

      const menuLocation = itemLocation ? `${itemLocation} > ${navElement}` : navElement;

      if (!Object.keys(subMenuItem).length) {

        let pageData = {

          pageName: navElement,

          pageLocation: menuLocation

        };

        menuItem.command = () => this.onNavMenuClick(pageData);

        if (!this.isInitialised && pageData.pageName.toLowerCase() == (this.sessionStorage.getData(StorageKeys.activeScreen) || "all claims").toLowerCase()) {

          this.isInitialised = !this.isInitialised;

          this.eventService.headerSelectionObservable.next(pageData);

        }

        configurationMenu.push(menuItem);

      } else {

        menuItem.items = [];

        configurationMenu.push(menuItem);

        this.navMenuConfiguration(subMenuItem, menuItem.items, menuLocation);

      }

    });

  }

  onNavMenuClick(pageData: any) {

    this.sessionStorage.setData(StorageKeys.activeScreen, pageData.pageName);

    this.eventService.headerSelectionObservable.next(pageData);

  }

  onPersonIconClick(event: Event) {

    this.menu.toggle(event);

  }

  logOut() {

    this.gloginService.userLogout();

  }

Directive to create dynamic form fields

    @Input() config: any;

    @Input() group: FormGroup;

    component: ComponentRef<Field>;

    constructor(

        private resolver: ComponentFactoryResolver,

        private container: ViewContainerRef

    ){}

    ngOnChanges() {

        if (this.component) {

            this.component.instance.config = this.config;

            this.component.instance.group = this.group;

        }

    }

    ngOnInit() {

        if (!components[this.config.type]) {

            const supportedType = Object.keys(components).join(', ');

            throw new Error(

                `Trying to use an unsupported type (${this.config.type}).

                Supported types: ${supportedType}`

            );

        }

        const component = this.resolver.resolveComponentFactory<Field>(components[this.config.type]);

        this.component = this.container.createComponent(component);

        this.component.instance.config = this.config;

        this.component.instance.group = this.group;

    }