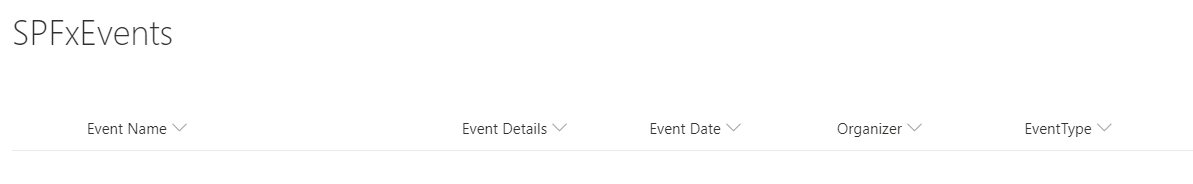
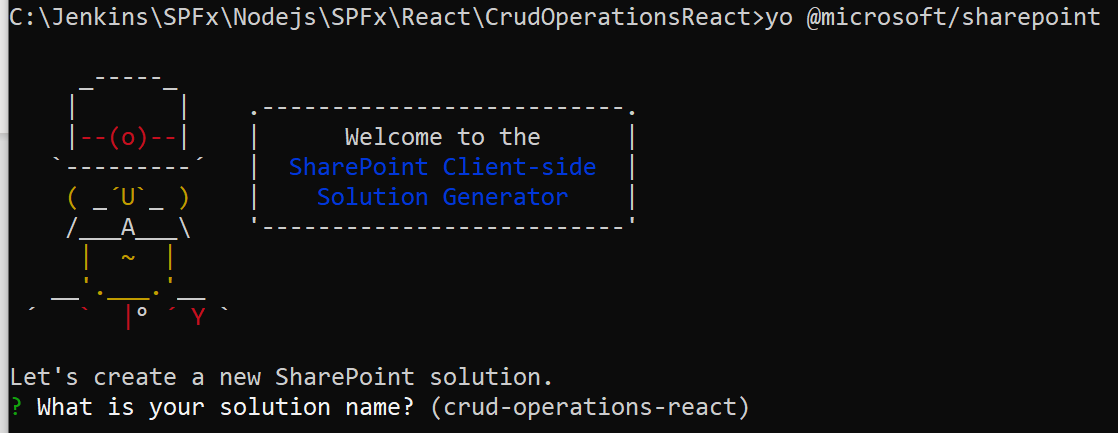
**Create a react curd operations - Save SharePoint list form**

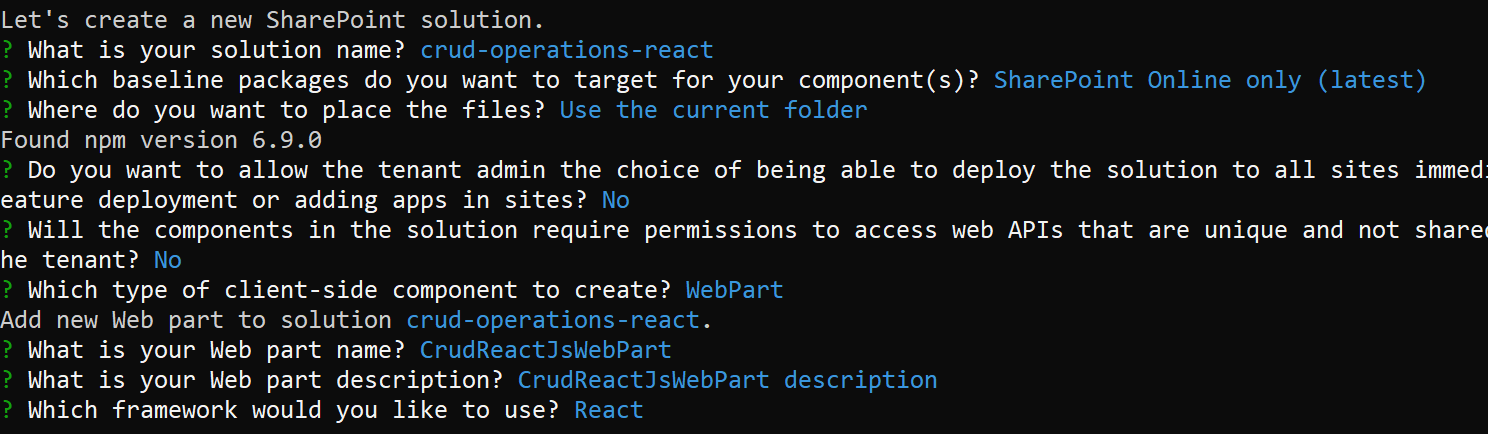
1. Create a SharePoint list “SPFxEvents” with below columns <https://jpower4mvp.sharepoint.com/sites/SPFxTraining/Lists/SPFxEvents/AllItems.aspx>
   1. Title – already available – rename to Event Name
   2. Event Date – date
   3. Organizer – people & group
   4. Event Details – Multiline
   5. Event Type - Choice

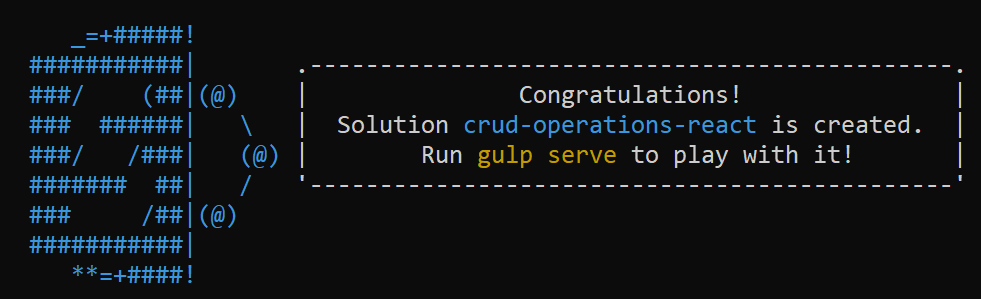


1. Create a Spfx webpart
   1. Open Command Prompt
   2. Create a folder **CrudOperationsReact**
   3. Navigate to **CrudOperationsReact** folder
   4. Create a new web part by running the Yeoman SharePoint Generator
   5. yo @microsoft/sharepoint



* 1. When prompt fill all information’s





Install fluent ui

* **npm install @fluentui/react**

Open the project in visual studio code using > code .

Create new file under component IListItem.ts and add below code

export interface IListItem{

  Id:number;

  Title?:string;

  EventDate?:string;

  Organizer?:string;

  EventDetails?:string;

  EventType?:string;

}

Then go to ICrudReactJsWebPartProps.ts and replace with below code

import {SPHttpClient} from '@microsoft/sp-http';

export interface ICurdOperationProps {

  description: string;

  siteUrl:string;

  spHttpClient:SPHttpClient;

}

Then create new file under component folder ICrudReactJsWebPartState.ts to manage the state and add below code

import { IListItem } from './IListItem';

export interface ICurdOperationState{

  spItems:IListItem[];

  showNewForm:boolean;

}

Create a new react component under component folder SPListItems.tsx and add below code

import \* as React from 'react';

import {IListItem} from './IListItem';

import { List } from '@fluentui/react/lib/List';

export interface IListItemsProps{

  spItems:IListItem[];

}

export class SPListItems extends React.Component<IListItemsProps,{}>{

  public render():React.ReactElement<IListItemsProps>{

    return(

      <div>

        <List items={this.props.spItems} onRenderCell={this.onRenderCell}/>

        <ul>

          {/\*

            this.props.spItems.map(spitem =>(

              <li>

                ID: {spitem.Id} - Event Name : {spitem.Title} - EventDate : {spitem.EventDate} - EventType: {spitem.EventType}

              </li>

            ))

            \*/}

        </ul>

        </div>

    );

  }

  private onRenderCell = (items:IListItem, index:number |undefined):JSX.Element=>

  {

    return(<div>

      {items.Id} {items.Title} {items.EventDetails} {items.EventDate} {items.Organizer} {items.EventType}

    </div>);

  }

}

Create a new react component under component folder NewForm.tsx and add below code

import \* as React from 'react';

import {SPHttpClient,SPHttpClientResponse} from '@microsoft/sp-http';

import {TextField} from '@fluentui/react/lib/TextField';

import { DatePicker, DayOfWeek, IDatePickerStrings, mergeStyleSets, PrimaryButton } from '@fluentui/react';

import { IPersonaProps } from '@fluentui/react/lib/Persona';

import { CompactPeoplePicker, IBasePickerSuggestionsProps, ValidationState } from '@fluentui/react/lib/Pickers';

import { Dropdown, DropdownMenuItemType, IDropdownStyles, IDropdownOption } from '@fluentui/react/lib/Dropdown';

import { assign } from '@microsoft/sp-lodash-subset';

import {IListItem} from './IListItem';

export interface INewFormProps

{

  siteUrl:string;

  spHttpClient:SPHttpClient;

  maxNrOfUsers?:number;

}

const suggestionProps: IBasePickerSuggestionsProps={

  suggestionsHeaderText: 'Sugessted People'

};

const limitedSearchAdditionalProps:IBasePickerSuggestionsProps={

  searchForMoreText: 'Search For More'

};

const options: IDropdownOption[] =[

  {key :'Conference', text:'Conference'},

  {key :'Training', text:'Training'},

  {key :'Town Hall Meeting', text:'Town Hall Meeting'},

];

const dropdownStyles: Partial<IDropdownStyles> = {

  dropdown: { width: 300 },

};

export interface INewFormState{

  Title?:string;

  EventDate?:Date;

  Organizer?:string;

  EventDetails?:string;

  EventType?:string;

  UserID?:string;

}

const limitedSearchSuggestionProps:IBasePickerSuggestionsProps = assign(limitedSearchAdditionalProps,suggestionProps);

export class NewForm extends React.Component<INewFormProps,INewFormState,{}>{

  private listItemEntityTypeName: string = undefined;

  constructor(props:INewFormProps,state:INewFormState)

  {

    super(props);

    this.state={

      Title:"",

      EventDate:null,

      Organizer:"",

      EventDetails:"",

      EventType:"",

      UserID:'0'

    };

  }

  public render():React.ReactElement<INewFormProps>

  {

    return(

      <div>

      <div>Welcome to New Form</div>

      <label>Event Name</label>

      <TextField

        id="txt\_eventname"

        title={this.state.Title}

        placeholder="Please enter event name..."

        onChange={(event,value)=>{this.setState({Title:value});}}

        />

        <label>Event Details</label>

      <TextField

        id="txt\_eventdetails"

        title={this.state.EventDetails}

        placeholder="Please enter event details..."

        multiline rows={5}

        onChange={(event,value)=>{this.setState({EventDetails:value});}}

        />

        <label>Event Date</label>

        <DatePicker

        id="dt\_eventdate"

        placeholder="Select a date"

        onSelectDate={date=> this.setState({EventDate:date})}

        />

        <label>Organizer</label>

        <CompactPeoplePicker

        onResolveSuggestions={this.onResolveSuggestions}

        pickerSuggestionsProps={limitedSearchSuggestionProps}

        className={'ms-PeoplePicker'}

        itemLimit={1}

        inputProps={{

          onBlur: (ev: React.FocusEvent<HTMLInputElement>) => console.log('onBlur called'),

          onFocus: (ev: React.FocusEvent<HTMLInputElement>) => console.log('onFocus called'),

          'aria-label': 'People Picker',

        }}

        resolveDelay={300}

        onChange={(this.\_onChangePeoplePicker.bind(this))}

      />

        <label>Event Type</label>

        <Dropdown id="drd\_eventtype"

        placeholder="Select the Event Type"

        options={options}

        styles={dropdownStyles}

        onChange={(event,value)=>{this.setState({EventType:value.text});}}

        />

        <label></label>

        <PrimaryButton text="Save" onClick={this.\_SaveNewItem.bind(this)}/>

      </div>

    );

  }

  private \_onChangePeoplePicker = (items?:any) =>{

    let loginname ="";

    let UserID:string = "";

    items.map(item=>(

      loginname = item.itemID

    ));

    this.\_getUserID(loginname);

  }

private \_getUserID(userAccountName):void{

  if(userAccountName)

  {

    userAccountName = "'" + userAccountName + "'";

    userAccountName = encodeURIComponent(userAccountName);

    const url:string = `${this.props.siteUrl}/\_api/web/siteusers(@v)?@v=${userAccountName}`;

      console.log(url);

      this.props.spHttpClient.get(url,SPHttpClient.configurations.v1,

          {

            headers: {

              'Accept': 'application/json;odata=verbose',

              'odata-version': ''

            }

          })

      .then((response:SPHttpClientResponse)=>{

        return response.json();

      },(error:any):void=>{

        console.log('error');

      })

      .then((jsonresponse:any)=>{

        let userid:string =  jsonresponse.d.Id;

        this.setState(

          {Organizer:userid}

          );

      });

  }

}

private \_SaveNewItem(event):void{

  let Title = this.state.Title;

  let EventDetails = this.state.EventDetails;

  let EventDate = this.state.EventDate;

  let EventType = this.state.EventType;

  let Organizer = this.state.Organizer;

  this.getListItemEntityTypeName()

      .then((listItemEntityTypeName: string): Promise<SPHttpClientResponse> => {

        const body: string = JSON.stringify({

          '\_\_metadata':

          {

            'type': listItemEntityTypeName

          },

          'Title': Title,

          'EventDetails':EventDetails,

          'OrganizerId':Number(Organizer), //{'results':[Organizer]},

          'EventDate': EventDate.toISOString(),

          //'EventType': {"\_\_metadata":{"type":"Collection(Edm.String)"},"results":[EventType]}  // multi selector

          'EventType': EventType

        });

        return this.props.spHttpClient.post(`${this.props.siteUrl}/\_api/web/lists/getbytitle('SPFxEvents')/items`,

          SPHttpClient.configurations.v1,

          {

            headers: {

              'Accept': 'application/json;odata=verbose',

              'Content-type': 'application/json;odata=verbose',

              'odata-version': ''

            },

            body: body

          });

      })

      .then((response: SPHttpClientResponse): Promise<IListItem> => {

        return response.json();

      });

}

  //Get Peoples to list

  private onResolveSuggestions = (searchText: string, currentPersonas: IPersonaProps[]): IPersonaProps[] | Promise<IPersonaProps[]> => {

    return this.onFilterChanged(searchText, currentPersonas, this.props.maxNrOfUsers);

  }

  private onFilterChanged = (filterText: string, currentPersonas: IPersonaProps[], limitResults?: number): IPersonaProps[] | Promise<IPersonaProps[]> => {

    return new Promise<IPersonaProps[]>((resolve,reject) => {

        let filteredPersonas: IPersonaProps[] = [];

    this.props.spHttpClient.get(`${this.props.siteUrl}/\_api/search/query?querytext='\*${filterText}\*'&rowlimit=10&sourceid='b09a7990-05ea-4af9-81ef-edfab16c4e31'`,

    SPHttpClient.configurations.v1,

    {

      headers: {

        'Accept': 'application/json;odata=nometadata',

        'odata-version': ''

      }

    })

    .then((response:SPHttpClientResponse):Promise<{PrimaryQueryResult:any}> =>{

      return response.json();

    })

    .then((response:{PrimaryQueryResult:any}):void =>{

        let revelantResults:any = response.PrimaryQueryResult.RelevantResults;

        let resultCount:number = revelantResults.TotalRows;

        if(resultCount>0)

        {

          revelantResults.Table.Rows.forEach( (row)=>{

            let persona:IPersonaProps ={};

            row.Cells.forEach((cell)=>{

              if(cell.Key === 'JobTitle')

              persona.secondaryText = cell.Value;

              if(cell.Key === 'PictureURL')

              persona.imageUrl = cell.Value;

              if(cell.Key === 'PreferredName')

              persona.primaryText = cell.Value;

              if(cell.Key === 'AccountName')

              persona.itemID = cell.Value;

            });

            filteredPersonas.push(persona);

          });

        }

       resolve (filteredPersonas);

      },(error:any):void =>{

        reject();

    });

  });

  }

  private getListItemEntityTypeName(): Promise<string> {

    return new Promise<string>((resolve: (listItemEntityTypeName: string) => void, reject: (error: any) => void): void => {

      if (this.listItemEntityTypeName) {

        resolve(this.listItemEntityTypeName);

        return;

      }

      this.props.spHttpClient.get(`${this.props.siteUrl}/\_api/web/lists/getbytitle('SPFxEvents')?$select=ListItemEntityTypeFullName`,

        SPHttpClient.configurations.v1,

        {

          headers: {

            'Accept': 'application/json;odata=nometadata',

            'odata-version': ''

          }

        })

        .then((response: SPHttpClientResponse): Promise<{ ListItemEntityTypeFullName: string }> => {

          return response.json();

        }, (error: any): void => {

          reject(error);

        })

        .then((response: { ListItemEntityTypeFullName: string }): void => {

          this.listItemEntityTypeName = response.ListItemEntityTypeFullName;

          resolve(this.listItemEntityTypeName);

        });

    });

  }

}

Go to CurdOperation.tsx file and add below headers

import { ICurdOperationState } from './ICurdOperationState';

import {IListItem} from './IListItem';

import {SPListItems} from './SPListItems';

import {PrimaryButton} from '@fluentui/react/lib/Button';

import {NewForm} from './NewForm';

import {SPHttpClient, SPHttpClientResponse} from '@microsoft/sp-http';

extends props and state to the class and add a constructor

export default class CurdOperation extends React.Component<ICurdOperationProps,ICurdOperationState, {}> {

  constructor(props:ICurdOperationProps, state:ICurdOperationState)

  {

    super(props);

    this.state={

      spItems:[],

      showNewForm:false

    };

  }

Add componentWillMount

public componentWillMount():void{

    this.getItems()

    .then((\_items:IListItem[])=>{

      this.setState({

        spItems:\_items

      });

    });

  }

Update the render control and add two react components

public render(): React.ReactElement<ICurdOperationProps> {

    return (

      <div className={ styles.curdOperation }>

        <div className={ styles.container }>

          <div className={ styles.row }>

            <div className={ styles.column }>

              <div className={`ms-grid-row`}>

              <span className={ styles.title }>Crud Operations</span>

              <SPListItems spItems={this.state.spItems} />

              </div>

            </div>

            </div>

            <div className={ styles.row }>

            <div className={ styles.column }>

            <div>

               <PrimaryButton text="New Item" onClick={this.showNewForm.bind(this)}/>

               {this.state.showNewForm? <NewForm siteUrl={this.props.siteUrl} spHttpClient={this.props.spHttpClient}/>: null }

            </div>

            </div>

            </div>

          </div>

      </div>

    );

  }

Hide and show New Form

private showNewForm(event):void{

    this.setState({

      showNewForm:true

    });

  }

**Get data from SharePoint**

private getItems():Promise<IListItem[]>{

  return new Promise<IListItem[]>((resolve,reject)=>{

    const url:string = `${this.props.siteUrl}/\_api/lists/getbytitle('SPFxEvents')/items?select=Id,Title,Event Date,Organizer,Event Details,Event Type`;

      console.log(url);

      this.props.spHttpClient.get(url,SPHttpClient.configurations.v1,

        {

          headers: {

            'Accept': 'application/json;odata=nometadata',

            'odata-version': ''

          }

        })

    .then((response:SPHttpClientResponse)=>{

      return response.json();

    },(error:any):void=>{

      reject(error);

    }

    )

    .then((jsonresponse:any)=>{

      let splistItems:IListItem[]=[];

      console.log(jsonresponse.value.length);

      if(jsonresponse.value.length ==0)

      {

        console.log('No records found');

      }

      else{

        console.log('Count : '+ jsonresponse.value.length);

      }

      for(let i=0;i<jsonresponse.value.length;i++)

      {

        splistItems.push({

        Id:jsonresponse.value[i].Id,

        Title:jsonresponse.value[i].Title,

        EventDate:jsonresponse.value[i].EventDate,

        Organizer:jsonresponse.value[i].Organizer,

        EventDetails:jsonresponse.value[i].EventDetails,

        EventType:jsonresponse.value[i].EventType

        });

        resolve(splistItems);

      }

    });

  });

}