Go to src => webparts => your file name => components and create following files for interface,

1. **PnpCrudWebPart.ts**
2. **PnpCrud.module.scss**
3. **IPnpCrudState.ts**
4. **IPnpCrudProps.ts**
5. **PnpCrud.tsx**

**PnpCrudWebPart.ts**

import \* as React from 'react';

import \* as ReactDom from 'react-dom';

import { Version } from '@microsoft/sp-core-library';

import {

  IPropertyPaneConfiguration,

  PropertyPaneTextField

} from '@microsoft/sp-property-pane';

import { BaseClientSideWebPart } from '@microsoft/sp-webpart-base';

import { IReadonlyTheme } from '@microsoft/sp-component-base';

import \* as strings from 'SpfxMfrpWebPartStrings';

import SpfxMfrp from './components/SpfxMfrp';

import { ISpfxMfrpProps } from './components/ISpfxMfrpProps';

import { sp } from "@pnp/sp/presets/all";

export interface ISpfxMfrpWebPartProps {

  description: string;

}

export default class SpfxMfrpWebPart extends BaseClientSideWebPart<ISpfxMfrpWebPartProps> {

  private \_isDarkTheme: boolean = false;

  private \_environmentMessage: string = '';

  protected onInit(): Promise<void>

  {​​​​​​this.\_environmentMessage = this.\_getEnvironmentMessage();

    return super.onInit().then(\_ =>{​​​​​​sp.setup({

    ​​​​​​spfxContext: this.context}​​​​​​);

  }​​​​​​);

}​​​​​​

  public render(): void {

    const element: React.ReactElement<ISpfxMfrpProps> = React.createElement(

      SpfxMfrp,

      {

        description: this.properties.description,

        isDarkTheme: this.\_isDarkTheme,

        environmentMessage: this.\_environmentMessage,

        hasTeamsContext: !!this.context.sdks.microsoftTeams,

        userDisplayName: this.context.pageContext.user.displayName

      }

    );

    ReactDom.render(element, this.domElement);

  }

  private \_getEnvironmentMessage(): string {

    if (!!this.context.sdks.microsoftTeams) { // running in Teams

      return this.context.isServedFromLocalhost ? strings.AppLocalEnvironmentTeams : strings.AppTeamsTabEnvironment;

    }

    return this.context.isServedFromLocalhost ? strings.AppLocalEnvironmentSharePoint : strings.AppSharePointEnvironment;

  }

  protected onThemeChanged(currentTheme: IReadonlyTheme | undefined): void {

    if (!currentTheme) {

      return;

    }

    this.\_isDarkTheme = !!currentTheme.isInverted;

    const {

      semanticColors

    } = currentTheme;

    this.domElement.style.setProperty('--bodyText', semanticColors.bodyText);

    this.domElement.style.setProperty('--link', semanticColors.link);

    this.domElement.style.setProperty('--linkHovered', semanticColors.linkHovered);

  }

  protected onDispose(): void {

    ReactDom.unmountComponentAtNode(this.domElement);

  }

  protected get dataVersion(): Version {

    return Version.parse('1.0');

  }

  protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {

    return {

      pages: [

        {

          header: {

            description: strings.PropertyPaneDescription

          },

          groups: [

            {

              groupName: strings.BasicGroupName,

              groupFields: [

                PropertyPaneTextField('description', {

                  label: strings.DescriptionFieldLabel

                })

              ]

            }

          ]

        }

      ]

    };

  }

}

**PnpCrud.module.scss**

@import '~office-ui-fabric-react/dist/sass/References.scss';.spfxMfrp {

  //Header

  .header {

  background-color: rgba($color: #91223a, $alpha: 1.0);

  color: white;

  float: left;

  padding-bottom: 5px;

  width: 100%;

  height: 70px;

  }

  .header img {

  float: left;

  margin-left: 30px;

  padding-right: 25px;

  }

  .heading{

  color: rgb(0, 0, 0);

  font-weight: bolder;

  margin: 17px;

  }

  //Footer

  .footer {

  background-color: rgb(240, 150, 173);

  height: 50px;

  padding-left: 30px;

  padding-top: 4px;

  //color: rgb(255, 255, 255);

  }

  .footing{

  color: black;

  font-weight: bolder;

  margin: 15px;

  }

  .container {

  max-width: 700px;

  margin: 0px auto;

  box-shadow: 0 2px 4px 0 rgba(0, 0, 0, 0.2), 0 25px 50px 0 rgba(0, 0, 0, 0.1);

  background-color: #238a9c;

  } .row {

  @include ms-Grid-row;

  @include ms-fontColor-white;

  //background-color: $ms-color-themeDark;

  padding: 20px;

  //background-color: rgb(175, 179, 236);

  } .column {

  @include ms-Grid-col;

  @include ms-lg10;

  @include ms-xl8;

  @include ms-xlPush2;

  @include ms-lgPush1;

  .add{

  background-color: #4767e6;

  border-color: #4767e6;

  }

  .delete{

  background-color: #e6474e;

  border-color: #e6474e;

  }

  .edit{

  background-color: #47e674;

  border-color: #47e674;

  }

  .reset{

  background-color: #474a48;

  border-color: #474a48;

  }

  } .title {

  @include ms-font-xl;

  @include ms-fontColor-white;

  } .subTitle {

  @include ms-font-l;

  @include ms-fontColor-white;

  } .description {

  @include ms-font-l;

  @include ms-fontColor-white;

  } .button {

  // Our button

  text-decoration: none;

  height: 32px; // Primary Button

  min-width: 80px;

  //background-color: $ms-color-themePrimary;

  //border-color: $ms-color-themePrimary;

  color: $ms-color-white; // Basic Button

  outline: transparent;

  position: relative;

  font-family: "Segoe UI WestEuropean","Segoe UI",-apple-system,BlinkMacSystemFont,Roboto,"Helvetica Neue",sans-serif;

  -webkit-font-smoothing: antialiased;

  font-size: $ms-font-size-m;

  font-weight: $ms-font-weight-regular;

  border-width: 0;

  text-align: center;

  cursor: pointer;

  display: inline-block;

  padding: 0 16px;

  .label {

  font-weight: $ms-font-weight-semibold;

  font-size: $ms-font-size-m;

  height: 32px;

  line-height: 32px;

  margin: 0 4px;

  vertical-align: top;

  display: inline-block;

  }

  }

  }

**IPnpCrud.State.ts**

import { Dropdown, DropdownMenuItemType, IDropdownStyles, IDropdownOption } from 'office-ui-fabric-react/lib/Dropdown';

export interface ISpfxMfrpState {

orderItems: IDropdownOption[];

customerItems: IDropdownOption[];

productItems: IDropdownOption[];

customerId: string;

productId: string;

orderId: any;

productType: string;

date: Date;

unitPrice: string;

numberOfUnits: string;

saleValue: string;

hideOrderId: boolean;

formateddate:string;

}

**IPnpCrud.Props.ts**

export interface ISpfxMfrpProps {

  description: string;

  isDarkTheme: boolean;

  environmentMessage: string;

  hasTeamsContext: boolean;

  userDisplayName: string;

}

**PnpCrud.tsx**

import \* as React from 'react';//import react library from package.json

import styles from './SpfxMfrp.module.scss';//styles

import { ISpfxMfrpProps } from './ISpfxMfrpProps';//importing interface

import { ISpfxMfrpState } from './SpfxMfrpState';// importing all the available states from SpfxMfrpState.ts

import { TextField } from 'office-ui-fabric-react'; // importing textfield from office ui

import { IStackTokens, Stack } from 'office-ui-fabric-react/lib/Stack';//import stack and stacktokens, provides padding between rows

import { Dropdown, IDropdownStyles, IDropdownOption } from 'office-ui-fabric-react/lib/Dropdown';// import dropdown from office ui

import { PrimaryButton } from 'office-ui-fabric-react';//import primary button from office ui

import { sp } from "@pnp/sp";//provides a fluent api for working with sharepoint REST

import "@pnp/sp/webs";//Webs serve as a container for lists, features, sub-webs, and all of the entity types.

import "@pnp/sp/lists";//list operations--add, get etc.

import "@pnp/sp/views";//defines the columns, ordering, and other details we see when we look at a list.

import "@pnp/sp/items";//get, add items from the list.

import { IItemAddResult } from '@pnp/sp/items';

//Declaration of dropdown array for required fields

var customerItems: IDropdownOption[] = [];

var productItems: IDropdownOption[] = [];

var orderItems: IDropdownOption[] = [];

//Declaration of required variables

var productName = '';

var customerName = '';

var orderId;

export default class SpfxMfrp extends React.Component<ISpfxMfrpProps, ISpfxMfrpState> {

  //Passing props and states to the constructor

  public constructor(props: ISpfxMfrpProps, state: ISpfxMfrpState) {

    super(props); //super is used to call the constructor of its parent class

    this.state = {

      //Assigned States from ISpfxMfrpState.tsx

      orderItems: [],

      orderId: '',

      customerItems: [],

      customerId: '',

      productItems: [],

      productType: '',

      date: new Date(),

      unitPrice: '',

      productId: '',

      numberOfUnits: '',

      saleValue: '',

      hideOrderId: false,

      formateddate:''

    };

    //Binding Functions

    // bind is used to send data as an arguments to the function  of class based components

    this.handleChange = this.handleChange.bind(this);

    this.handleChangeCust = this.handleChangeCust.bind(this);

    this.autoPopulate = this.autoPopulate.bind(this);

    this.handleUnitChange = this.handleUnitChange.bind(this);

    this.addToOrderList = this.addToOrderList.bind(this);

    this.handleChangeOrd = this.handleChangeOrd.bind(this);

    this.editOrderList = this.editOrderList.bind(this);

    this.deleteItem = this.deleteItem.bind(this);

    this.resetOrderList = this.resetOrderList.bind(this);

  }

  //initializing fields based on product name for autopopulate

  private handleChange(event): void {

    productName = event.key;

    this.setState({ numberOfUnits: '' }),

      this.setState({ saleValue: '' }),

      this.autoPopulate();

  }

  //Getting Vendor id when customer name is selected

  //async ensures that the function returns a promise. Other values are wrapped in a resolved promise automatically.

  //we used try/catch when we need to catch the error inside an event handler.

  //await makes javascript wait until that promise settles and returns its result.

  private async handleChangeCust(event): Promise<void> {

    try {

      customerName = event.key;

      let items = await sp.web.lists.getByTitle("Vendors").items.getPaged();//Getting the list items from the customers list

      for (let i = 0; i < items.results.length; i++) {

        if (items.results[i].VendorName == customerName) {

          this.setState({ customerId: items.results[i].VID});

        }

      }

    } catch (error) {

      console.error(error);

    }

  }

  //getting order details when OrderID is selected

  private async handleChangeOrd(event): Promise<void> {

    try {

      orderId = event.key;

      let items = await sp.web.lists.getByTitle("Orders").items.getPaged();//Getting the list items from the Orders list

      for (let i = 0; i < items.results.length; i++) {

        if (items.results[i].Order\_x0020\_ID == orderId) {

          this.setState({orderId: orderId});

          this.setState({ productId: items.results[i].ProductID});

          this.setState({ customerId: items.results[i].VendorID });

          this.setState({ numberOfUnits: items.results[i].UnitsSold});

          this.setState({ unitPrice: items.results[i].UnitPrice});

          this.setState({ saleValue: items.results[i].SaleValue});

        }

      }

      let customeritems = await sp.web.lists.getByTitle("Vendors").items.getPaged();

      for (let i = 0; i < customeritems.results.length; i++) {

        debugger;

        if (customeritems.results[i].VID == this.state.customerId) {

          customerName = customeritems.results[i].VendorName;

        }

      }

      let productitems = await sp.web.lists.getByTitle("Product").items.getPaged();

      for (let i = 0; i < productitems.results.length; i++) {

        if (productitems.results[i].PID == this.state.productId) {

          productName = productitems.results[i].ProductName;

          this.setState({ productType: productitems.results[i].ProductType });

          this.setState({ formateddate:String(productitems.results[i].ProductExpiryDate ).substring(0,10)});

        }

      }

    } catch (error) {

      console.error(error);

    }

  }

  //Autopopulating fields  when product name is selected

  private async autoPopulate(): Promise<void> {

    try {

      let items = await sp.web.lists.getByTitle("Product").items.getPaged();

      for (let i = 0; i < items.results.length; i++) {

        if (items.results[i].ProductName == productName) {

          this.setState({ productId: items.results[i].PID});

          this.setState({ productType: items.results[i].ProductType});

          this.setState({ unitPrice: items.results[i].ProductUnitPrice});

          this.setState({ formateddate:String(items.results[i].ProductExpiryDate ).substring(0,10)});

        }

      }

    } catch (error) {

      console.error(error);

    }

  }

  //Calculating Sale Value

  private handleUnitChange = (event) => {

    this.setState({ numberOfUnits: event.target.value.toString() });

    var units: number = parseInt(event.target.value);

    var unitPrice: number = parseInt(this.state.unitPrice);

    var calculate = units \* unitPrice;

    this.setState({ saleValue: calculate.toString() });

    return event;

  }

  //Adding order to orders list

  private async addToOrderList(event): Promise<void> {

    try {

      debugger;

      var unitvalid = this.state.numberOfUnits;

      if (customerName == "" || productName == "") {

        alert("Please Select Vendor Name From Dropdown" + "\n" + "Please Select Product Name From Dropdown");

      }

      else if (this.state.numberOfUnits == "" || this.state.numberOfUnits <= "0") {

        alert("Please Enter Number Of Units");

      }

      else if (Number(unitvalid) !== parseInt(unitvalid) && Number(unitvalid) % 1 !== 0) {

        alert("Please enter No. of Units as integer value");

      }

      else {

        debugger;

        let item = await sp.web.lists.getByTitle('Orders').items.add({

          VendorID: this.state.customerId.toString(),

          ProductID: this.state.productId.toString(),

          UnitsSold: +this.state.numberOfUnits,

          UnitPrice: +this.state.unitPrice,

          SaleValue: +this.state.saleValue,

          ProductName:productName,

          VendorName:customerName

          //Title: "title"

        });

        alert("Order is added successfully in the list.");

        this.resetOrderList();

      }

    } catch (error) {

      console.error(error);

    }

  }

  // //Editing orders

  private async editOrderList(event): Promise<void> {

    try {

      this.setState({ hideOrderId: true });

      var unitvalid = this.state.numberOfUnits;

      if (orderId == undefined) {

        alert("Select an order ID for editing");

      }

      else if (this.state.numberOfUnits == "" && this.state.customerId == "") {

        this.setState({ hideOrderId: true });

      }

      else {

        if (this.state.numberOfUnits == "" || this.state.numberOfUnits == "0") {

          alert("Please Enter Valid Number Of Units or You have entered zero units");

        }

        else if (Number(unitvalid) !== parseInt(unitvalid) && Number(unitvalid) % 1 !== 0) {

          alert("Please enter No. of Units as integer value");

        }

        else {

          let id: any = orderId;//from input

          id = id.replace(/[^\d]/g, '');  //Extracting only integer.

          id = parseInt(id, 10);         //Trimming Leading Zeros.

          if (id > 0) {

            let editOrderList = await sp.web.lists.getByTitle("Orders").items.getById(id).update({

            VendorID: this.state.customerId.toString(),

            ProductID: this.state.productId.toString(),

            UnitsSold: +this.state.numberOfUnits,

            UnitPrice: +this.state.unitPrice,

            SaleValue: +this.state.saleValue,

            ProductName:productName,

            VendorName:customerName

            //Title: "title"

          });

          alert("ORDERID -" + orderId + " is updated successfully in the list.");

          this.resetOrderList();

        }

        }

      }

    }

    catch (error) {

      console.error(error);

    }

  }

  //Deleting order

  private async deleteItem(event): Promise<void> {

    try {

      this.setState({ hideOrderId: true });

      if (orderId == undefined) {

        alert("Select an order ID to delete");

      }

      let id: any = orderId;//from input

      id = id.replace(/[^\d]/g, '');  //Extracting only integer.

      id = parseInt(id, 10);         //Trimming Leading Zeros.

      if (id > 0) {

        let deleteItem = await sp.web.lists.getByTitle("Orders").items.getById(id).delete();

        console.log(deleteItem);

        alert(`Item ID: ${orderId} deleted successfully!`);

        this.resetOrderList();

      }

      else {

        alert(`Please enter a valid Order id.`);

      }

    } catch (error) {

      console.log(error);

    }

  }

  //reset order list

  private resetOrderList(): void {

    customerName = '';

    productName = '';

    orderId = '';

    this.setState({

      date: new Date(),

      productType: '',

      unitPrice: '',

      numberOfUnits: '',

      saleValue: ''

    });

  }

  public async componentDidMount(): Promise<void> {

    // get all the items from a sharepoint list

    var reacthandler = this;

    sp.web.lists.getByTitle("Vendors").items.select('VendorName').get().then((data) => {

      for (var k in data) {

        customerItems.push({ key: data[k].VendorName, text: data[k].VendorName });

      }

      reacthandler.setState({ customerItems });

      console.log(customerItems);

      return customerItems;

    });

    sp.web.lists.getByTitle("Product").items.select('ProductName').get().then((data) => {

      for (var k in data) {

        productItems.push({ key: data[k].ProductName, text: data[k].ProductName });

      }

      reacthandler.setState({ productItems });

      console.log(productItems);

      return productItems;

    });

    sp.web.lists.getByTitle("Orders").items.select('Order\_x0020\_ID').get().then((data) => {

      for (var k in data) {

        orderItems.push({ key: data[k].Order\_x0020\_ID, text: data[k].Order\_x0020\_ID});

      }

      reacthandler.setState({ orderItems: orderItems });

      console.log(orderItems);

      return orderItems;

    });

  }

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

    const dropdownStyles: Partial<IDropdownStyles> = {

      dropdown: { width: 452 }

    };

    const stackTokens: IStackTokens = { childrenGap: 12 };

    return (

      <div className={styles.spfxMfrp}>

        <div className={styles.container}>

          <div>

            <div className={styles.header}>

              <header>

                <img src={require('./image/im6.jpg')} alt="Scar" width="65" height="40" />

                <h1 className={styles.heading}>Sales Order Form</h1>

              </header></div>

            <div className={styles.row}>

              <div className={styles.column}>

                <Stack tokens={stackTokens}>

                  <Dropdown

                    placeholder="Select Vendor Name"

                    label="Vendor Name"

                    selectedKey={customerName}

                    options={this.state.customerItems}

                    styles={dropdownStyles}

                    onChanged={this.handleChangeCust}

                  />

                </Stack>

                <Stack tokens={stackTokens}>

                  <Dropdown

                    placeholder="Select Product Name"

                    label="Product Name"

                    selectedKey={productName}

                    options={this.state.productItems}

                    styles={dropdownStyles}

                    onChanged={this.handleChange}

                  />

                </Stack>

                <TextField required={true}

                  placeholder="Product Type will Be Entered Automatically"

                  label="Product Type"

                  value={this.state.productType}

                  readOnly={true}

                  onChange={event => {

                    this.setState({ productType: this.state.productType });

                  }}

                /><br />

                <TextField label="Product Expiry Date"

                  placeholder=""

                  value={this.state.formateddate}

                  readOnly={true}

                />

                <TextField required={true}

                  placeholder="Product Unit Price"

                  label="Product Unit Price"

                  type="number"

                  value={this.state.unitPrice}

                  readOnly={true}

                  onChange={e => { this.setState({ unitPrice: this.state.unitPrice }); }}

                />

                <TextField required={true}

                  placeholder="Enter Number Of Units "

                  label="Number of units"

                  type="number"

                  value={this.state.numberOfUnits}

                  onChange={this.handleUnitChange}

                //onChange={e=>{this.setState({ numberOfUnits: 'e' })}}

                />

                <div className="HeadText">

                  <TextField required={true}

                    placeholder="Total Sale Value"

                    label="Sale Value"

                    type="number"

                    value={this.state.saleValue}

                    readOnly={true}

                    onChange={e => { this.setState({ saleValue: this.state.saleValue }); }}

                  />

                  <Stack tokens={stackTokens}>

                    {

                      this.state.hideOrderId ?

                        <Dropdown required={true}

                          placeholder="Select an Order ID to Edit or Delete"

                          label="Order ID"

                          selectedKey={orderId}

                          options={this.state.orderItems}

                          styles={dropdownStyles}

                          onChanged={this.handleChangeOrd} />

                        : null

                    }

                  </Stack><br></br>

                  <br></br>

                </div>

              </div>

              <div className={styles.column}>

                <hr />

                <PrimaryButton className={styles.add} onClick={this.addToOrderList} >ADD</PrimaryButton>&nbsp;&nbsp; {/\*non breaking space\*/}

                <PrimaryButton  className={styles.edit} onClick={this.editOrderList} >EDIT</PrimaryButton>&nbsp;&nbsp;

                <PrimaryButton className={styles.delete} onClick={this.deleteItem} >DELETE</PrimaryButton>&nbsp;&nbsp;

                <PrimaryButton className={styles.reset} onClick={this.resetOrderList}>RESET</PrimaryButton>&nbsp;&nbsp;

                <hr />

              </div>

            </div>

            <div className={styles.footer}>

              <footer>

                <section>

                  <h3 className={styles.footing}> &copy;Created By Shital@2022</h3>

                </section>

              </footer>

            </div>

          </div>

        </div>

      </div>

    );

  }

}