🏅

Air Schedule

How it works ?

▼ **CRUD**

- CRUD refers to the four basic operations a software application should be able to perform – Create, Read, Update, and Delete.
- In such apps, users must be able to **create data**, have access to the data in the UI by **reading** the data, **update** or **edit** the data, and **delete** the data.
- The API contains the code and methods, the database stores and helps the user retrieve the information, while the user interface helps users interact with the app.
- We can make crud operation in any programming languages.

Folder Structure➖

▼ Node Modules

- it contains all the npm packages which is we using in react application.

▼ public

- index.js

▼ src

- redux
- routes
- views
- app.js
- index.js
- service.js

Code With Description➖

▼ **node modules**

- *NPM* is short for node package manager
- it contains various already registered open source-packages.
- NPM command is **npm install**

▼ **public**

- Public folder contains static file such as **index.html , logo img, other assest which is not required.**
- index.html is simple html file contains one div with root id

▼ **src**

- This folder is the heart of react application as it contains javascript which needs to processed by **webpack.**

1. **Redux folder Structure**

▼ **redux**

- actions
- reducer
- types
- store

▼ **actions**

- action folder contains all the actions of react application which is done by users.

```
import { SET_SHEDULE, GET_SHEDULE, SHEDULE_ERROR } from "../types/users";
import fetchData from "../../service";
import { toast } from "react-toastify";

export const clearSheduleData = () => {
  return (dispatch) => {
    dispatch({ type: SHEDULE_ERROR, payload: {} });
  };
};

export const getSheduleData = () => {
  return (dispatch) => {
    let postData = {};
    postData.id = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cSchedule/getFullList",
      method: "GET",
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          let resultData = JSON.parse(result.data.response);
          dispatch({ type: GET_SHEDULE, payload: resultData });
        } else {
          dispatch({ type: SHEDULE_ERROR, payload: {} });
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};

export const getSingleSheduleData = (sheduleId) => {
  return (dispatch) => {
    let postData = {};
    postData.id = sheduleId;
    postData.userid = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cShedule/getEditData",
      method: "POST",
      data: { data: postData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === true
        ) {
          let resultData = JSON.parse(result.data.response);
          //dispatch({ type: GET_ORIGIN, payload: resultData });
        } else {
          dispatch({ type: SHEDULE_ERROR, payload: {} });
          //toast.error('Profile Updated Failed');
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};

export const saveSheduleData = (sheduleData) => {
  return (dispatch) => {
    let postData = sheduleData;
    postData.createdBy = sessionStorage.getItem("dokoid");

    console.log(postData);
    // debugger;
    // return;
    let response = fetchData({
      url: "/site/cSchedule/insert",
      method: "POST",
      data: { data: [postData] },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          toast.success("SHEDULE inserted Successfully");
          window.location.replace(window.location.origin + "/shedule/");
        } else {
          toast.error("SHEDULE Updated Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};

export const updateSheduleData = (sheduleData) => {
  return (dispatch) => {
    let postData = sheduleData;
    postData.createdBy = sessionStorage.getItem("dokoid");

    let response = fetchData({
      url: "/site/cSchedule/update",
      method: "POST",
      data: { data: [postData] },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          toast.success("SHEDULE Updated Successfully");
          window.location.replace(window.location.origin + "/shedule/");
        } else {
          toast.error("SHEDULE Updated Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};

export const multipleSaveSheduleData = (sheduleData) => {
  return (dispatch) => {
    let postData = sheduleData,
      modifiedPostData = [];

    for (var i = 0; i < postData.length; i += 1) {
      // postData[i]._userid = sessionStorage.getItem('dokoid');
      modifiedPostData.push(postData[i]);
    }

    let response = fetchData({
      url: "/site/cSchedule/insert",
      method: "POST",
      data: { data: modifiedPostData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          //dispatch({ type: SET_SHEDULE, payload: sheduleData });
          toast.success("SHEDULE Updated Successfully");
          window.location.replace(window.location.origin + "/shedule/");
        } else {
          toast.error("SHEDULE Updated Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};

export const deleteSheduleData = (deleteSheduleData) => {
  return (dispatch) => {
    let postData = deleteSheduleData;
    postData.createdBy = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cSchedule/deleteList",
      method: "POST",
      data: { data: postData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          dispatch(getSheduleData());
          toast.success("SHEDULE deleted Successfully");
        } else {
          toast.error("SHEDULE delete Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};

export const multipleDeleteSheduleData = (deleteSheduleData) => {
  return (dispatch) => {
    let postData = deleteSheduleData;
    postData._id = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cSchedule/deleteList",
      method: "POST",
      data: { data: postData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          dispatch(getSheduleData());
          toast.success("SHEDULE deleted Successfully");
        } else {
          toast.error("SHEDULE delete Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

Description

- Above javascript file is the **sheduleAction.js.**
- Above file contains eight actions like,

1. **clearSheduleData** action
    - in this action **SHEDULE_ERROR** type of action is dispatched.
    - **dispatch**
        1. **dispatch** is the redux function which is **Dispatches an action.**
        2. This is the only way to trigger the action.
        3. **clearSheduleData** function used to dispatch the action type of **SHEDULE_ERROR .**

1. **getSheduleData Action**

```
export const getSheduleData = () => {
  return (dispatch) => {
    let postData = {};
    postData.id = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cSchedule/getFullList",
      method: "GET",
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          let resultData = JSON.parse(result.data.response);
          dispatch({ type: GET_SHEDULE, payload: resultData });
        } else {
          dispatch({ type: SHEDULE_ERROR, payload: {} });
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- in this function two action are dispatched
    1. GET_SHEDULE
    2. SHEDULE_ERROR
- **getSheduleData** in this function API is callled
    - if in response **result.data.status === 1** comes then **GET_SHEDULE** this action will be dispatched otherwise
    - **SHEDULE_ERROR** action will be dispatched.

3. **updateSheduleData** action.

```
export const updateSheduleData = (sheduleData) => {
  return (dispatch) => {
    let postData = sheduleData;
    postData.createdBy = sessionStorage.getItem("dokoid");

    let response = fetchData({
      url: "/site/cSchedule/update",
      method: "POST",
      data: { data: [postData] },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          toast.success("SHEDULE Updated Successfully");
          window.location.replace(window.location.origin + "/shedule/");
        } else {
          toast.error("SHEDULE Updated Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- this funcation is used to update the pincode value .
- in this function POST call api is used.
- post data format is the Array of object.
- The **sessionStorage** object let you store key/value pairs in the browser.
- **getItem** is used to retrive object from **sessionStorage** and we can use object in our component.

```
window.location.replace(window.location.origin + "/shedule/");
```

```
```

- this line is used to replace current **url** resource with provided url.

4. **getSingleSheduleData** action

```
export const getSingleSheduleData = (sheduleId) => {
  return (dispatch) => {
    let postData = {};
    postData.id = sheduleId;
    postData.userid = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cShedule/getEditData",
      method: "POST",
      data: { data: postData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === true
        ) {
          let resultData = JSON.parse(result.data.response);
          //dispatch({ type: GET_ORIGIN, payload: resultData });
        } else {
          dispatch({ type: SHEDULE_ERROR, payload: {} });
          //toast.error('Profile Updated Failed');
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- **getSingleSheduleData** function is triggered to get sheduleData
- in this function ‘cShedule/getFullList’ api is called with **GET** Method.

5. **saveSheduleData** action

```
export const saveSheduleData = (sheduleData) => {
  return (dispatch) => {
    let postData = sheduleData;
    postData.createdBy = sessionStorage.getItem("dokoid");

    console.log(postData);
    // debugger;
    // return;
    let response = fetchData({
      url: "/site/cSchedule/insert",
      method: "POST",
      data: { data: [postData] },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          toast.success("SHEDULE inserted Successfully");
          window.location.replace(window.location.origin + "/shedule/");
        } else {
          toast.error("SHEDULE Updated Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- **saveSheduleData** function is triggered to insert pincode Data to FullList API
- in this function ‘cSchedule/insert’ api is called with **POST** Method.
- The **hasOwnProperty()** method returns a boolean indicating whether the object has the specified property as its own property.

6. **multipleSaveSheduleData action**

```
export const multipleSaveSheduleData = (sheduleData) => {
  return (dispatch) => {
    let postData = sheduleData,
      modifiedPostData = [];

    for (var i = 0; i < postData.length; i += 1) {
      // postData[i]._userid = sessionStorage.getItem('dokoid');
      modifiedPostData.push(postData[i]);
    }

    let response = fetchData({
      url: "/site/cSchedule/insert",
      method: "POST",
      data: { data: modifiedPostData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          //dispatch({ type: SET_SHEDULE, payload: sheduleData });
          toast.success("SHEDULE Updated Successfully");
          window.location.replace(window.location.origin + "/shedule/");
        } else {
          toast.error("SHEDULE Updated Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- in this function API POST method called.
- this function is used to save multiple data in api
- in post data we given object Id.

7. **deleteSheduleData action**

```
export const deleteSheduleData = (deleteSheduleData) => {
  return (dispatch) => {
    let postData = deleteSheduleData;
    postData.createdBy = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cSchedule/deleteList",
      method: "POST",
      data: { data: postData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          dispatch(getSheduleData());
          toast.success("SHEDULE deleted Successfully");
        } else {
          toast.error("SHEDULE delete Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- this function is used to delete shedule Data from api .
- in this function delete api is used with post call.
- toast is used to show notification messege.

8. **multipleDeleteSheduleData action**

```
export const multipleDeleteSheduleData = (deleteSheduleData) => {
  return (dispatch) => {
    let postData = deleteSheduleData;
    postData._id = sessionStorage.getItem("dokoid");
    let response = fetchData({
      url: "/site/cSchedule/deleteList",
      method: "POST",
      data: { data: postData },
    });
    response
      .then((result) => {
        if (
          result.hasOwnProperty("data") === true &&
          result.data.status === 1
        ) {
          dispatch(getSheduleData());
          toast.success("SHEDULE deleted Successfully");
        } else {
          toast.error("SHEDULE delete Failed");
        }
      })
      .catch((e) => {
        console.log(e);
        toast.error(e);
      });
  };
};
```

- this function is used to delete multiple pincode Data from api .
- this function carrying **deleteSheduleData** parameter which is contains unique id with perticular pincode data.

▼ **reducer**

- this folder contains all the reducer files with combineReducer function
- **combineReducers** is used to reduce the function into single reducing function.

```
import {GET_SHEDULE, SET_SHEDULE, SHEDULE_ERROR} from '../types/users';

const initialState = {
    shedule: [],
    listData: [],
    loading:true
}


function sheduleReducer(state = initialState, action) {
    switch (action.type) {
        case GET_SHEDULE:
            //const addHSN = [...state.HSN, action.payload];
            return {
                ...state,
                shedule:[...action.payload],
                loading:false
            }
        case SET_SHEDULE:
            //const addHSN = [...state.HSN, action.payload];
            return {
                ...state,
                shedule:action.payload,
                loading:false
            }

        case SHEDULE_ERROR:
            //const addHSN = [...state.HSN, action.payload];
            return {
                ...state,
                shedule: action.payload,
                loading:false
            }


        default:
            return state
    }
}

export default sheduleReducer;
```

- GET_SHEDULE, SET_SHEDULE, SHEDULE_ERROR this type of action are dispatched in this js file.
- in reducer file reducer function comes with initialstate, that means we have to declare a initialstate first.
- in this reducer function we used **switch method .**
- switch method is executing function based on action type.

▼ **index.js**

```
import { combineReducers } from "redux";
import userReducer from "./userReducers";
import profileReducer from "./profileReducers";
import hsnReducers from "./hsnReducers";
import originReducers from "./originReducers";
import destinationReducers from "./destinationReducers";
import lclOriginReducers from "./lclOriginReducers";
import lclDestinationReducers from "./lclDestinationReducers";
import sezReducers from "./sezReducers";
import sheduleReducers from "./sheduleReducers";
import lclSheduleReducers from "./lclSheduleReducers";
import originPickReducers from "./originPickReducers";
import pincodeReducer from "./pincodeReducer";
import rateReducer from "./rateReducer";
import lclRateReducer from "./lclRateReducer";
import currencyRducer from "./currencyReducer";
import originPartnerReducer from "./originPartnerReducer";
import operationTeamReducer from "./operationTeamReducer";
import myBookingReducer from "./myBookingReducers";
import customerReducer from "./customerReducer";
import lclPickReducer from "./lclPickReducer";
import newBookingReducer from "./newBookingReducer";
import portReducer from "./portReducer";
import adminReducer from "./adminReducer";
import emailReducer from "./emailReducer";
import consoleReducer from "./consoleReducer";
import holidayReducer from "./holidayReducer";

export default combineReducers({
  usersList: userReducer,
  profileData: profileReducer,
  hsnData: hsnReducers,
  originDatas: originReducers,
  destinationDatas: destinationReducers,
  sezDatas: sezReducers,
  sheduleDatas: sheduleReducers,
  originPickDatas: originPickReducers,
  pincodeDatas: pincodeReducer,
  rateDatas: rateReducer,
});
```

- This file is used to combine the all reducer function in single reducing function.
- The **combineReducers** helper function turns an object whose values are different reducing functions into a single reducing function you can pass to **createStore.**
- **combineReducers** is simply **a utility function to simplify the most common use case when writing Redux reducers**
  .
  .

▼ **store folder**

  - **index.js**

```
import { applyMiddleware, configureStore } from '@reduxjs/toolkit'
import thunk from 'redux-thunk'
//import { composeWithDevTools } from 'redux-devtools-extension'
import { persistStore, persistReducer } from 'redux-persist'
import storage from 'redux-persist/lib/storage'

import rootReducer from '../reducer'

const persistConfig = {
  key: 'doko',
  storage,
}

const persistedReducer = persistReducer(persistConfig, rootReducer)

const initalState = {}

const middleware = [thunk]

// const store = configureStore(
//   persistedReducer,
//   initalState,
//   composeWithDevTools(applyMiddleware(...middleware))
// )

const store = configureStore({
  reducer: persistedReducer,
  initalState: initalState,
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware(
      {
        serializableCheck: false,
      },
      applyMiddleware(...middleware)
    ),
})

const persistor = persistStore(store)

export { persistor }
export default store
```

- this file contains application store which is accesible in entire components.
- **Middleware i**s the suggested way to extend Redux with custom functionality.
- **applyMiddleware** which allows us to use custom middleware as well as Redux middlewares like redux-thunk.
- **redux-thunk** is a type of middleware.
- **configureStore** is used to create **Store.**
- r**edux-persist -** Redux Persist is a popular library which lets you add persistence to the store.
- `persistReducer` has a general purpose "migrate" config which will be called after getting stored state but before actually reconciling with the reducer.

▼ types folder

- in **types** folder we created one **user.js** file. which contains all type of actions.

```
export const SET_SHEDULE = "SET_SHEDULE";
export const EDIT_SHEDULE = "EDIT_SHEDULE";
export const GET_SHEDULE = "GET_SHEDULE";
export const DELETE_SHEDULE = "DELETE_SHEDULE";
export const SHEDULE_ERROR = "SHEDULE_ERROR";
export const CLEAR_SHEDULE = "CLEAR_SHEDULE";
```

- this folder contains the type of action .

▼ routes folder

- **routes** folder contains **index.js** file.

▼ index.js

```
import { lazy } from "react";

const SheduleComponent = lazy(() => import("../views/Shedule/list"));
const AddSheduleComponent = lazy(() => import("../views/Shedule/Add"));
const EditSheduleComponent = lazy(() => import("../views/Shedule/Edit"));

const routs = [
  {
    path: "shedule",
    component: SheduleComponent,
    exact: true,
    private: true,
  },
  {
    path: "shedule/add",
    component: AddSheduleComponent,
    exact: true,
    private: true,
  },
  {
    path: "shedule/edit/:id",
    component: EditSheduleComponent,
    exact: true,
    private: true,
  }
];

export default routs;
```

- in this file we given all **route** of our react application.
- if the url **path** matches with given path then only releted component will be rendered.

▼ **views folder**

- in this view folder we created all our react application components.

▼ **Shedule**

1. **add.js**
2. **edit.js**
3. **list.js**

- **code with description as belows.**

1. **add.js**

```
import React, { useState, useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import Header from "../Common/header";
import Sidebar from "../Common/sidebar";
import { saveSheduleData } from "../../redux/actions/sheduleAction";
import { getOriginData } from "../../redux/actions/originAction";
import { getDestinationData } from "../../redux/actions/destinationAction";
import "./index.css";
import { toast } from "react-toastify";
import {
  consoleValidation,
  btbValidation,
} from "../Validation/scheduleValidation";
import { useForm } from "react-hook-form";
import { yupResolver } from "@hookform/resolvers/yup";
import { useNavigate } from "react-router-dom";

function Add(params) {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  var consoleStyleData,
    b2bStyleData = "";
  var originData = useSelector((state) => state.originDatas);
  var destinationData = useSelector((state) => state.destinationDatas);
  const [consoleFlag, setConsoleFlag] = useState(true);
  const [b2bFlag, setB2bFlag] = useState(false);
  const [checked, setChecked] = useState(false);

  const createdBy = sessionStorage.getItem("dokoid");

  var [sheduleData, setSheduleData] = useState({
    // cSheduleID: "",
    cShedulepol: "",
    cShedulepod: "",
    cSheduleRouting: "",
    cSheduleTransit: "",
    cSheduleFrequency: "",
    // cSheduleBookingCutoff: "",
    // cShedulecargoHandoverCutoff: "",
    // cSheduleCoolingPeriod: "",
    cBookingCutoffFormat: "",
    cHandoverCutoffFormat: "",
    cvalidUntil: "",
    cSheduleType: "1",
    cSheduleETA: "",
    cSheduleETD: "",
    cSheduleHeightLimit: "",
    cSStatus: "1",
    createdBy: createdBy,
  });

  console.log(sheduleData);

  const setSheduleProcessData = (name) => {
    return ({ target: { value } }) => {
      if (name == "cSheduleType") {
        if (value == "1") {
          setConsoleFlag(true);
          setB2bFlag(false);
        }
        if (value == "2") {
          setConsoleFlag(false);
          setB2bFlag(true);
        }
      }

      setSheduleData((oldValues) => ({
        ...oldValues,
        [name]: value,
        createdBy: createdBy,
      }));
    };
  };
  console.log(sheduleData);

  // console.log(checkbox);

  useEffect(() => {
    dispatch(getOriginData());
    dispatch(getDestinationData());
  }, []);

  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm({
    resolver: async (data, context, options) => {
      // btbValidation you can debug your validation schema here
      console.log("formData", data);
      //console.log('validation result', await yupResolver(consoleValidation)(data, context, options))
      if (data.cSheduleType === "1") {
        return yupResolver(consoleValidation)(data, context, options);
      } else {
        return yupResolver(btbValidation)(data, context, options);
      }
    },
    //,
    //resolver: yupResolver(consoleValidation),
  });

  var allCheckBoxes = [];
```

```
    const checkBoxes = document.getElementsByName("Frequency");
    for (var i = 0; i < checkBoxes.length; i++) {
      if (checkBoxes[i].checked === true) {
        const checkboxValue = checkBoxes[i].value;
        allCheckBoxes.push(checkboxValue);
      }

      let scheduleFrequency = allCheckBoxes.toString();

      sheduleData["cSheduleFrequency"] = scheduleFrequency;
    }

    const saveShedule = async () => {
      let selectedType = document.getElementById("cSheduleType").value;
      const btbValid = await btbValidation.isValid(sheduleData);
      const consoleValid = await consoleValidation.isValid(sheduleData);
      debugger;
      if (selectedType === "2") {
        if (btbValid === true) {
          dispatch(saveSheduleData(sheduleData));
        } else {
          toast.error("Please fill All field Data");
        }
      } else if (selectedType === "1") {
        console.log(consoleValid);
        if (consoleValid === true && allCheckBoxes.length > 0) {
          dispatch(saveSheduleData(sheduleData));
        } else {
          toast.error("Select frequency");
        }
      } else {
        console.log("Please select any Type");
      }
    };

    if (consoleFlag === false) {
      consoleStyleData = {};
    } else {
      consoleStyleData = { display: "none" };
    }

    if (b2bFlag === false) {
      b2bStyleData = {};
    } else {
      b2bStyleData = { display: "none" };
    }

    return (
      <main id="main" className="main">
        <Header />
        <Sidebar />
        <section className="section profile">
          <div className="row">
            <div className="col-xl-8">
              <div className="card">
                <div className="card-body pt-3">
                  <div className="tab-content pt-2">
                    <div
                      className="tab-pane fade show active profile-edit pt-3"
                      id="profile-edit"
                    >
                      <form onSubmit={handleSubmit(saveShedule)}>
                        <div className="row mb-3">
                          <label className="col-md-4 col-lg-3 col-form-label">
                            Type
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <select
                              className="form-select"
                              defaultValue={"1"}
                              name="cSheduleType"
                              {...register("cSheduleType", {
                                value: sheduleData.cSheduleType,
                                onChange: setSheduleProcessData("cSheduleType"),
                              })}
                              aria-label="Default select example"
                              id="cSheduleType"
                            >
                              {/* <option value="">Select Type</option> */}
                              <option value="1">Console</option>
                              {/* <option value="2">back-2-back</option> */}
                            </select>
                            <p className="scheduleError">
                              {errors?.cSheduleType?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3">
                          <label
                            htmlFor="cShedulepol"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            POL
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <select
                              className="form-select"
                              defaultValue={""}
                              name="cShedulepol"
                              {...register("cShedulepol", {
                                value: sheduleData.cShedulepol,
                                onChange: setSheduleProcessData("cShedulepol"),
                              })}
                            >
                              <option value="">Select POL</option>
                              {originData.origin.map((e, index) => {
                                return (
                                  <option value={e._id} key={index}>
                                    {e.oName}
                                  </option>
                                );
                              })}
                            </select>
                            <p className="scheduleError">
                              {errors?.cShedulepol?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3">
                          <label
                            htmlFor="cShedulepod"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            POD
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <select
                              className="form-select"
                              defaultValue={""}
                              name="cShedulepod"
                              {...register("cShedulepod", {
                                value: sheduleData.cShedulepod,
                                onChange: setSheduleProcessData("cShedulepod"),
                              })}
                            >
                              <option value="">Select POD</option>
                              {destinationData.destination.map((e, index) => {
                                return (
                                  <option value={e._id} key={index}>
                                    {e.dName}
                                  </option>
                                );
                              })}
                            </select>
                            <p className="scheduleError">
                              {errors?.cShedulepod?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3">
                          <label
                            htmlFor="cSheduleRouting"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            Routing
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <input
                              name="cSheduleRouting"
                              {...register("cSheduleRouting", {
                                value: sheduleData.cSheduleRouting,
                                onChange:
                                  setSheduleProcessData("cSheduleRouting"),
                              })}
                              type="text"
                              required=""
                              className="form-control"
                              id="cSheduleRouting"
                            />
                            <p className="scheduleError">
                              {errors?.cSheduleRouting?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3">
                          <label
                            htmlFor="cSheduleTransit"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            Transit
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <input
                              name="cSheduleTransit"
                              {...register("cSheduleTransit", {
                                value: sheduleData.cSheduleTransit,
                                onChange:
                                  setSheduleProcessData("cSheduleTransit"),
                              })}
                              type="text"
                              required=""
                              className="form-control"
                              id="cSheduleTransit"
                            />

                            <p className="scheduleError">
                              {errors?.cSheduleTransit?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3" style={b2bStyleData}>
                          <label
                            htmlFor="cSheduleFrequency"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            Frequency
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <div className="d-flex justify-content-start align-items-center gap-2">
                              <label htmlFor="1">1</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="1"
                                id="1"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                              <label htmlFor="2">2</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="2"
                                id="2"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                              <label htmlFor="3">3</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="3"
                                id="3"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                              <label htmlFor="4">4</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="4"
                                id="4"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                              <label htmlFor="5">5</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="5"
                                id="5"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                              <label htmlFor="6">6</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="6"
                                id="6"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                              <label htmlFor="7">7</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="7"
                                id="7"
                                onClick={(e) => setChecked(e.target.value)}
                              />
                            </div>

                            <p className="scheduleError">
                              {errors?.cSheduleFrequency?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3" style={consoleStyleData}>
                          <label
                            htmlFor="cSheduleETA"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            ETA
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <input
                              name="cSheduleETA"
                              {...register("cSheduleETA", {
                                value: sheduleData.cSheduleETA,
                                onChange: setSheduleProcessData("cSheduleETA"),
                              })}
                              type="date"
                              required=""
                              className="form-control"
                              id="cSheduleETA"
                            />
                            <p className="scheduleError">
                              {errors?.cSheduleETA?.message}
                            </p>
                          </div>
                        </div>
                        <div className="row mb-3" style={consoleStyleData}>
                          <label
                            htmlFor="cSheduleETD"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            ETD
                          </label>
                          <div className="col-md-8 col-lg-9">
                            <input
                              name="cSheduleETD"
                              {...register("cSheduleETD", {
                                value: sheduleData.cSheduleETD,
                                onChange: setSheduleProcessData("cSheduleETD"),
                              })}
                              type="date"
                              required=""
                              className="form-control"
                              id="cSheduleETD"
                            />
                            <p className="scheduleError">
                              {errors?.cSheduleETD?.message}
                            </p>
                          </div>
                        </div>

                        <div className="row mb-3">
                          <label
                            htmlFor="cSheduleHeightLimit"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            Height Limit
                          </label>
                          <div className="col-md-8 col-lg-9">
```

```
            <input
              name="cSheduleHeightLimit"
              {...register("cSheduleHeightLimit", {
                value: sheduleData.cSheduleHeightLimit,
                onChange: setSheduleProcessData(
                  "cSheduleHeightLimit"
                ),
              })}
              type="text"
              required=""
              className="form-control"
              id="cSheduleHeightLimit"
            />
            <p className="scheduleError">
              {errors?.cSheduleHeightLimit?.message}
            </p>
          </div>
        </div>

        <div className="row mb-3" style={b2bStyleData}>
          <label
            htmlFor="cvalidUntil"
            className="col-md-4 col-lg-3 col-form-label"
          >
            Valid Until
          </label>
          <div className="col-md-8 col-lg-9">
            <input
              name="cvalidUntil"
              value={sheduleData.cvalidUntil}
              {...register("cvalidUntil", {
                onChange: setSheduleProcessData("cvalidUntil"),
              })}
              type="date"
              required=""
              className="form-control"
              id="cvalidUntil"
            />
            <p className="scheduleError">
              {errors?.cvalidUntil?.message}
            </p>
          </div>
        </div>

        <div className="row mb-3">
          <label
            htmlFor="cBookingCutoffFormat"
            className="col-md-4 col-lg-3 col-form-label"
          >
            Booking Cut Off Format
          </label>
          <div className="col-md-8 col-lg-9">
            <input
              name="cBookingCutoffFormat"
              {...register("cBookingCutoffFormat", {
                value: sheduleData.cBookingCutoffFormat,
                onChange: setSheduleProcessData(
                  "cBookingCutoffFormat"
                ),
              })}
              type="text"
              required=""
              className="form-control"
              id="cBookingCutoffFormat"
            />
            <p className="scheduleError">
              {errors?.cBookingCutoffFormat?.message}
            </p>
          </div>
        </div>

        <div className="row mb-3">
          <label
            htmlFor="cHandoverCutoffFormat"
            className="col-md-4 col-lg-3 col-form-label"
          >
            HandOver Cut Off Format
          </label>
          <div className="col-md-8 col-lg-9">
            <input
              name="cHandoverCutoffFormat"
              {...register("cHandoverCutoffFormat", {
                value: sheduleData.cHandoverCutoffFormat,
                onChange: setSheduleProcessData(
                  "cHandoverCutoffFormat"
                ),
              })}
              type="text"
              required=""
              className="form-control"
              id="cHandoverCutoffFormat"
            />
            <p className="scheduleError">
              {errors?.cHandoverCutoffFormat?.message}
            </p>
          </div>
        </div>

        <div className="row mb-3">
          <label className="col-md-4 col-lg-3 col-form-label">
            Status
          </label>
          <div className="col-md-8 col-lg-9">
            <select
              className="form-select"
              defaultValue={sheduleData.cSStatus}
              {...register("cSStatus", {
                value: sheduleData.cSStatus,
                onChange: setSheduleProcessData("cSStatus"),
              })}
              aria-label="Default select example"
            >
              <option value="">Select Status</option>
              <option value="1">Active</option>
              <option value="2">In-Active</option>
            </select>
            <p className="scheduleError">
              {errors?.cSStatus?.message}
            </p>
          </div>
        </div>

        <div className="text-center d-flex gap-1 justify-content-end">
          <button
            className="btn btn-danger"
            onClick={() => navigate("/shedule")}
          >
            Cancel
          </button>
          <button type="submit" className="btn btn-primary">
            Save Changes
          </button>
        </div>
      </form>
```

```
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </section>
    </main>
  );
}

export default Add;
```

▼ **Description**

1. **saveShedule** is the onSubmit function used to save data to db.
2. formState contains **errors** object which is belongs to error messeges . with the help of only we are showing error messeges in UI of applications.
3. **useDispatch** is imported from **'react-redux'** which is used to dispatch the function.
4. toast is imported from **'react-toast'** which is used to toast notification in app.
5. **useform** and **yupResolver** are used to give validation to the input field in this app component.
6. in this file component we have imported **sheduleValidation** file from validation folder. which is used to check the validation first.
7. **async** makes a function return a Promise and **await** makes a function wait for a Promise.
8. **handleSubmit** is triggered when the given validation is successfully proceed.
9. **isValid** is the yupResolver property used to check the validation with object state.
10. **setSheduleProcessData** is the onchange function used to get onchange value from that perticular input field.
11. **useEffect** is used to perform the side effect on component , its normaly used to fetching data , timer function , for directly update in dom.
12. **useState** is used to store the state value . it helps us to track the state in function component.

    ```
                                 {...register("cSStatus", {
                                  value: sheduleData.cSStatus,
                                  onChange: setSheduleProcessData("cSStatus"),
                                })}
    ```

13. register method allows you to register an input or select element and apply validation rules to React Hook Form. Validation rules are all based on the HTML standard and also allow for custom validation methods.
14. if this `register` invokes it receive following methods
    a. onChange
    b. onBlur
    c. ref
    d. name
    e. value

2. **edit.js**

```
import React, { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import Header from "../Common/header";
import Sidebar from "../Common/sidebar";
import {
  updateSheduleData,
  getSheduleData,
} from "../../redux/actions/sheduleAction";
import { useParams } from "react-router-dom";
import { getOriginData } from "../../redux/actions/originAction";
import { getDestinationData } from "../../redux/actions/destinationAction";
import { toast } from "react-toastify";
import {
  btbValidation,
  consoleValidation,
} from "../Validation/scheduleValidation";
import "./index.css";
import { useForm } from "react-hook-form";
import { yupResolver } from "@hookform/resolvers/yup";

function Edit(params) {
  const dispatch = useDispatch();
  var consoleStyleData,
    b2bStyleData = "";
  let { id } = useParams();
  var i = 0;

  var originData = useSelector((state) => state.originDatas);
  var destinationData = useSelector((state) => state.destinationDatas);
  const [consoleFlag, setConsoleFlag] = useState(true);
  const [b2bFlag, setB2bFlag] = useState(false);
  const [checked, setChecked] = useState([]);

  const createdBy = sessionStorage.getItem("dokoid");

  var [sheduleData, setSheduleData] = useState({
    // cSheduleID: "",
    cShedulepol: "",
    cShedulepod: "",
    cSheduleRouting: "",
    cSheduleTransit: "",
    cSheduleFrequency: "",
    // cSheduleBookingCutoff: "",
    // cShedulecargoHandoverCutoff: "",
    // cSheduleCoolingPeriod: "",
    cBookingCutoffFormat: "",
    cHandoverCutoffFormat: "",
    cvalidUntil: "",
    cSheduleType: "",
    cSheduleETA: "",
    cSheduleETD: "",
    cSheduleHeightLimit: "",
    cSStatus: "",
    createdBy: createdBy,
  });

  var sheduleSelectorData = useSelector((state) => state.sheduleDatas);
  const setSheduleProcessData = (name) => {
    return ({ target: { value } }) => {
      if (name == "cSheduleType") {
        if (value == "1") {
          setConsoleFlag(true);
          setB2bFlag(false);
        }

        if (value == "2") {
          setConsoleFlag(false);
          setB2bFlag(true);
        }
      }
      setSheduleData((oldValues) => ({ ...oldValues, [name]: value }));
    };
  };

   useEffect(() => {
        // console.log(id);
    if (sheduleSelectorData.shedule.length == 0) {
       dispatch(getSheduleData);
     }

      if (sheduleSelectorData.shedule.length > 0) {
       for (; i < sheduleSelectorData.shedule.length; i += 1) {
         var singleSheduleData = sheduleSelectorData.shedule;
         if (singleSheduleData[i]._id == id) {
           setChecked(
            ...checked,
             singleSheduleData[i].cSheduleFrequency.split(",")
          );

           if (singleSheduleData[i].cSheduleType == "1") {
             setConsoleFlag(true);
             setB2bFlag(false);
          }

          if (singleSheduleData[i].cSheduleType == "2") {
             setConsoleFlag(false);
            setB2bFlag(true);
          }
          setSheduleData({
            // cSheduleID: singleSheduleData[i].cSheduleID,
            cShedulepol: singleSheduleData[i].cShedulepol,
            cShedulepod: singleSheduleData[i].cShedulepod,
            cSheduleRouting: singleSheduleData[i].cSheduleRouting,
            cSheduleTransit: singleSheduleData[i].cSheduleTransit,
            cBookingCutoffFormat:singleSheduleData[i].cBookingCutoffFormat,
            cHandoverCutoffFormat:singleSheduleData[i].cHandoverCutoffFormat,
            cvalidUntil:singleSheduleData[i].cvalidUntil,
            // cSheduleBookingCutoff: singleSheduleData[i].cSheduleBookingCutoff,
            // cShedulecargoHandoverCutoff:
            //    singleSheduleData[i].cShedulecargoHandoverCutoff,
            // cSheduleCoolingPeriod: singleSheduleData[i].cSheduleCoolingPeriod,
            cSheduleType: singleSheduleData[i].cSheduleType,
          cSheduleETA: singleSheduleData[i].cSheduleETA,
          cSheduleETD: singleSheduleData[i].cSheduleETD,
          cSheduleHeightLimit: singleSheduleData[i].cSheduleHeightLimit,
          cSStatus: singleSheduleData[i].cSStatus,
        });

         reset(singleSheduleData[i]);
      }
    }
  }

    if (originData.origin.length == 0) {
      dispatch(getOriginData());
     }
     if (destinationData.destination.length == 0) {
       dispatch(getDestinationData());
     }
  }, []);

  const {
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm({
    resolver: async (data, context, options) => {
      // btbValidation you can debug your validation schema here
      console.log("formData", data);
      //console.log('validation result', await yupResolver(consoleValidation)(data, context, options))
      if (data.cSheduleType === "1") {
        return yupResolver(consoleValidation)(data, context, options);
      } else {
        return yupResolver(btbValidation)(data, context, options);
      }
    },
    defaultValues: sheduleData,
    //,
    //resolver: yupResolver(consoleValidation),
  });

  //  var cSheduleBookingCutoff= moment(new Date, 'DD-MM-YYYY').format();
  //     cSheduleBookingCutoff=cSheduleBookingCutoff.split('/')[0];

  console.log(sheduleData);
  console.log(checked);

  const saveShedule = async () => {
    sheduleData["cSheduleFrequency"] = checked
       .filter((e) => e !== "")
       .join(",");
    let selectedType = document.getElementById("cSheduleType").value;
    const btbValid = await btbValidation.isValid(sheduleData);
    const consoleValid = await consoleValidation.isValid(sheduleData);

    if (selectedType === "1") {
      console.log(consoleValid);
      if (consoleValid === true) {
        sheduleData.id = id;
        console.log(consoleValid);
        dispatch(updateSheduleData(sheduleData));
      } else {
        toast.error("Please fill All field Data");
      }
    } else if (selectedType === "2") {
      if (btbValid === true) {
        sheduleData.id = id;
        console.log(sheduleData);
        dispatch(updateSheduleData(sheduleData));
      } else {
        toast.error("Please Fill All Field Data");
      }
    }
    {
      console.log("Please select any Type");
    }
  };

  if (consoleFlag === false) {
    consoleStyleData = {};
  } else {
    consoleStyleData = { display: "none" };
  }

  if (b2bFlag === false) {
    b2bStyleData = {};
  } else {
    b2bStyleData = { display: "none" };
  }
  console.log(b2bFlag);
  return (
    <main id="main" className="main">
      {/* {console.log(checked)} */}
      <Header />
      <Sidebar />
      <section className="section profile">
        <div className="row">
          <div className="col-xl-8">
            <div className="card">
              <div className="card-body pt-3">
                <div className="tab-content pt-2">
                  <div
                    className="tab-pane fade show active profile-edit pt-3"
                    id="profile-edit"
                  >
                    <form onSubmit={handleSubmit(saveShedule)}>
                      <div className="row mb-3">
                        <label className="col-md-4 col-lg-3 col-form-label">
                          Type
                        </label>
                        <div className="col-md-8 col-lg-9">
                          <select
                            className="form-select"
                            name="cSheduleType"
                            {...register("cSheduleType", {
                              value: sheduleData.cSheduleType,
                              onChange: setSheduleProcessData("cSheduleType"),
                            })}
                            aria-label="Default select example"
                            id="cSheduleType"
                          >
                            {/* <option value="">Select Type</option> */}
                            <option value="1">Console</option>
                            {/* <option value="2">back-2-back</option> */}
                          </select>
                          <p className="scheduleError">
                            {errors?.cSheduleType?.message}
                          </p>{" "}
                        </div>
                      </div>
                      {/* <div className="row mb-3">
                        <label
                          htmlFor="cSheduleId"
                          className="col-md-4 col-lg-3 col-form-label"
                        >
                          Shedule Id
                        </label>
                        <div className="col-md-8 col-lg-9">
                          <input
                            name="cSheduleID"
                            type="number"
                            {...register("cSheduleID", {
                              value: sheduleData.cSheduleID,
                              onChange: setSheduleProcessData("cSheduleID"),
                            })}
                            required=""
                            className="form-control"
                            id="cSheduleID"
                          />
                          <p className="scheduleError">
                            {errors?.cSheduleID?.message}
                          </p>
                        </div>
                      </div> */}
                      <div className="row mb-3">
                        <label
                          htmlFor="cShedulepol"
                          className="col-md-4 col-lg-3 col-form-label"
                        >
                          POL
                        </label>
                        <div className="col-md-8 col-lg-9">
                          <select
                            className="form-select"
                            value={sheduleData.cShedulepol}
                            {...register("cShedulepol", {
                              value: sheduleData.cShedulepol,
                              onChange: setSheduleProcessData("cShedulepol"),
                            })}
                          >
                            <option value="">Select POL</option>
                            {originData.origin.map((e, index) => {
                              return (
                                <option value={e._id} key={index}>
                                  {e.oName}
                                </option>
                              );
                            })}
                          </select>
                          <p className="scheduleError">
                            {errors?.cShedulepol?.message}
                          </p>
                        </div>
                      </div>

                      <div className="row mb-3">
                        <label
                          htmlFor="cShedulepod"
                          className="col-md-4 col-lg-3 col-form-label"
                        >
                          POD
                        </label>
                        <div className="col-md-8 col-lg-9">
                          <select
                            className="form-select"
                            value={sheduleData.cShedulepod}
                            {...register("cShedulepod", {
                              value: sheduleData.cShedulepod,
                              onChange: setSheduleProcessData("cShedulepod"),
                            })}
                          >
                            <option value="">Select POD</option>
```

```
                  {destinationData.destination.map((e, index) => {
                    return (
                      <option value={e._id} key={index}>
                        {e.dName}
                      </option>
                    );
                  })}
                </select>
                <p className="scheduleError">
                  {errors?.cShedulepod?.message}
                </p>
              </div>
            </div>

            <div className="row mb-3">
              <label
                htmlFor="cSheduleRouting"
                className="col-md-4 col-lg-3 col-form-label"
              >
                Routing
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cSheduleRouting"
                  {...register("cSheduleRouting", {
                    value: sheduleData.cSheduleRouting,
                    onChange:
                      setSheduleProcessData("cSheduleRouting"),
                  })}
                  type="text"
                  required=""
                  className="form-control"
                  id="cSheduleRouting"
                />
                <p className="scheduleError">
                  {errors?.cSheduleRouting?.message}
                </p>
              </div>
            </div>

            <div className="row mb-3">
              <label
                htmlFor="cSheduleTransit"
                className="col-md-4 col-lg-3 col-form-label"
              >
                Transit
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cSheduleTransit"
                  {...register("cSheduleTransit", {
                    value: sheduleData.cSheduleTransit,
                    onChange:
                      setSheduleProcessData("cSheduleTransit"),
                  })}
                  type="text"
                  required=""
                  className="form-control"
                  id="cSheduleTransit"
                />
                <p className="scheduleError">
                  {errors?.cSheduleTransit?.message}
                </p>
              </div>
            </div>

            <div className="row mb-3" style={b2bStyleData}>
```

```
                          <label
                            htmlFor="cSheduleFrequency"
                            className="col-md-4 col-lg-3 col-form-label"
                          >
                            Frequency
                          </label>

                            {/* <input
                              name="cSheduleFrequency"
                              {...register("cSheduleFrequency", {
                                value: sheduleData.cSheduleFrequency,
                                onChange:
                                  setSheduleProcessData("cSheduleFrequency"),
                              })}
                              type="text"
                              required=""
                              className="form-control"
                              id="cSheduleFrequency"
                            /> */}
                            {/* <p className="scheduleError">
                              {errors?.cSheduleFrequency?.message}
                            </p>
                            </div> */}
                           <div className="col-md-8 col-lg-9">
                            <div className="d-flex justify-content-start align-items-center gap-2">
                              <label htmlFor="1">1</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="1"
                                id="1"
                                checked={checked.includes("1") ? "checked" : ""}
                                onClick={(e) =>
                                  e.target.checked === true
                                    ? setChecked([...checked, "1"])
                                    : setChecked(checked.filter((e) => e != "1"))
                                }
                              />
                              <label htmlFor="2">2</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="2"
                                id="2"
                                checked={checked.includes("2") ? "checked" : ""}
                                onClick={(e) =>
                                  e.target.checked === true
                                    ? setChecked([...checked, "2"])
                                    : setChecked(checked.filter((e) => e != "2"))
                                }
                              />
                              <label htmlFor="3">3</label>
                              <input
                                name="Frequency"
                                type="checkbox"
                                value="3"
                                id="3"
                                checked={checked.includes("3") ? "checked" : ""}
                                onClick={(e) =>
                                  e.target.checked === true
                                    ? setChecked([...checked, "3"])
                                    : setChecked(checked.filter((e) => e != "3"))
                                }
                              />
                              <label htmlFor="4">4</label>
                              <input
                                name="Frequency"
```

```
        type="checkbox"
        value="4"
        id="4"
        checked={checked.includes("4") ? "checked" : ""}
        onClick={(e) =>
          e.target.checked === true
            ? setChecked([...checked, "4"])
            : setChecked(checked.filter((e) => e != "4"))
        }
      />
      <label htmlFor="5">5</label>
      <input
        name="Frequency"
        type="checkbox"
        value="5"
        id="5"
        checked={checked.includes("5") ? "checked" : ""}
        onClick={(e) =>
          e.target.checked === true
            ? setChecked([...checked, "5"])
            : setChecked(checked.filter((e) => e != "5"))
        }
      />
      <label htmlFor="6">6</label>
      <input
        name="Frequency"
        type="checkbox"
        value="6"
        id="6"
        checked={checked.includes("6") ? "checked" : ""}
        onClick={(e) =>
          e.target.checked === true
            ? setChecked([...checked, "6"])
            : setChecked(checked.filter((e) => e != "6"))
        }
      />
      <label htmlFor="7">7</label>
      <input
        name="Frequency"
        type="checkbox"
        value="7"
        id="7"
        checked={checked.includes("7") ? "checked" : ""}
        onClick={(e) =>
          e.target.checked === true
            ? setChecked([...checked, "7"])
            : setChecked(checked.filter((e) => e != "7"))
        }
      />
    </div>
  </div>
</div>

{/* <div className="row mb-3" style={b2bStyleData}>
  <label
    htmlFor="cSheduleBookingCutoff"
    className="col-md-4 col-lg-3 col-form-label"
  >
    Booking Cut Off
  </label>
  <div className="col-md-8 col-lg-9">
    <input
      name="cSheduleBookingCutoff"
      {...register("cSheduleBookingCutoff", {
        value: sheduleData.cSheduleBookingCutoff,
        onChange: setSheduleProcessData(
          "cSheduleBookingCutoff"
```

```
            ),
          })}
          type="date"
          required=""
          className="form-control"
          id="cSheduleBookingCutoff"
        />
        <p className="scheduleError">
          {errors?.cSheduleBookingCutoff?.message}
        </p>
      </div>
    </div>

    <div className="row mb-3" style={b2bStyleData}>
      <label
        htmlFor="cShedulecargoHandoverCutoff"
        className="col-md-4 col-lg-3 col-form-label"
      >
        Cargo Hand Over Cutoff
      </label>
      <div className="col-md-8 col-lg-9">
        <input
          name="cShedulecargoHandoverCutoff"
          {...register("cShedulecargoHandoverCutoff", {
            value: sheduleData.cShedulecargoHandoverCutoff,
            onChange: setSheduleProcessData(
              "cShedulecargoHandoverCutoff"
            ),
          })}
          type="date"
          required=""
          className="form-control"
          id="cShedulecargoHandoverCutoff"
        />
        <p className="scheduleError">
          {errors?.cShedulecargoHandoverCutoff?.message}
        </p>
      </div>
    </div>

    <div className="row mb-3" style={b2bStyleData}>
      <label
        htmlFor="cSheduleCoolingPeriod"
        className="col-md-4 col-lg-3 col-form-label"
      >
        Cooling Period
      </label>
      <div className="col-md-8 col-lg-9">
        <input
          name="cSheduleCoolingPeriod"
          {...register("cSheduleCoolingPeriod", {
            value: sheduleData.cSheduleCoolingPeriod,
            onChange: setSheduleProcessData(
              "cSheduleCoolingPeriod"
            ),
          })}
          type="date"
          required=""
          className="form-control"
          id="cSheduleCoolingPeriod"
        />
        <p className="scheduleError">
          {errors?.cSheduleCoolingPeriod?.message}
        </p>
      </div>
    </div> */}
```

```
            <div className="row mb-3" style={consoleStyleData}>
              <label
                htmlFor="cSheduleETA"
                className="col-md-4 col-lg-3 col-form-label"
              >
                ETA
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cSheduleETA"
                  {...register("cSheduleETA", {
                    value: sheduleData.cSheduleETA,
                    onChange: setSheduleProcessData("cSheduleETA"),
                  })}
                  type="date"
                  required=""
                  className="form-control"
                  id="cSheduleETA"
                />
                <p className="scheduleError">
                  {errors?.cSheduleETA?.message}
                </p>
              </div>
            </div>

            <div className="row mb-3" style={consoleStyleData}>
              <label
                htmlFor="cSheduleETD"
                className="col-md-4 col-lg-3 col-form-label"
              >
                ETD
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cSheduleETD"
                  {...register("cSheduleETD", {
                    value: sheduleData.cSheduleETD,
                    onChange: setSheduleProcessData("cSheduleETD"),
                  })}
                  type="date"
                  required=""
                  className="form-control"
                  id="cSheduleETD"
                />
                <p className="scheduleError">
                  {errors?.cSheduleETD?.message}
                </p>
              </div>
            </div>
            <div className="row mb-3">
              <label
                htmlFor="cSheduleHeight"
                className="col-md-4 col-lg-3 col-form-label"
              >
                Height Limit
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cSheduleHeightLimit"
                  {...register("cSheduleHeightLimit", {
                    value: sheduleData.cSheduleHeightLimit,
                    onChange: setSheduleProcessData(
                      "cSheduleHeightLimit"
                    ),
                  })}
                  type="text"
                  required=""
                  className="form-control"
                  id="cSheduleHeightLimit"
                />
                <p className="scheduleError">
                  {errors?.cSheduleHeightLimit?.message}
                </p>
              </div>
            </div>
            <div className="row mb-3">
              <label
                htmlFor="cBookingCutoffFormat"
                className="col-md-4 col-lg-3 col-form-label"
              >
                Booking Cut Off Format
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cBookingCutoffFormat"
                  {...register("cBookingCutoffFormat", {
                    value: sheduleData.cBookingCutoffFormat,
                    onChange: setSheduleProcessData(
                      "cBookingCutoffFormat"
                    ),
                  })}
                  type="text"
                  required=""
                  className="form-control"
                  id="cBookingCutoffFormat"
                />
                <p className="scheduleError">
                  {errors?.cBookingCutoffFormat?.message}
                </p>
              </div>
            </div>
            <div className="row mb-3">
              <label
                htmlFor="cHandoverCutoffFormat"
                className="col-md-4 col-lg-3 col-form-label"
              >
                Handover Cut Off Format
              </label>
              <div className="col-md-8 col-lg-9">
                <input
                  name="cHandoverCutoffFormat"
                  {...register("cHandoverCutoffFormat", {
                    value: sheduleData.cHandoverCutoffFormat,
                    onChange: setSheduleProcessData(
                      "cHandoverCutoffFormat"
                    ),
                  })}
                  type="text"
                  required=""
                  className="form-control"
                  id="handoverCutoffFormat"
                />
                <p className="scheduleError">
                  {errors?.cHandoverCutoffFormat?.message}
                </p>
              </div>
            </div>
            <div className="row mb-3">
              <label
                htmlFor="cvalidUntil"
                className="col-md-4 col-lg-3 col-form-label"
              >
                Valid Until
              </label>
```

```
                        <div className="col-md-8 col-lg-9">
                          <input
                            name="cvalidUntil"
                            {...register("cvalidUntil", {
                              value: sheduleData.cvalidUntil,
                              onChange: setSheduleProcessData("cvalidUntil"),
                            })}
                            type="date"
                            required=""
                            value={sheduleData.cvalidUntil}
                            className="form-control"
                            id="cvalidUntil"
                          />
                          <p className="scheduleError">
                            {errors?.cvalidUntil?.message}
                          </p>
                        </div>
                      </div>
                      <div className="row mb-3">
                        <label className="col-md-4 col-lg-3 col-form-label">
                          Status
                        </label>
                        <div className="col-md-8 col-lg-9">
                          <select
                            className="form-select"
                            value={sheduleData.cSStatus}
                            name="cSStatus"
                            {...register("cSStatus", {
                              value: sheduleData.cSStatus,
                              onChange: setSheduleProcessData("cSStatus"),
                            })}
                            aria-label="Default select example"
                          >
                            <option value="">Select Status</option>
                            <option value="1">Active</option>
                            <option value="2">In-Active</option>
                          </select>
                          <p className="scheduleError">
                            {errors?.cSStatus?.message}
                          </p>
                        </div>
                      </div>

                      <div className="text-center">
                        <button type="submit" className="btn btn-primary">
                          Save Changes
                        </button>
                      </div>
                    </form>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </section>
    </main>
  );
}

export default Edit;
```

▼ **Description**

1. `useSelector` is imported from **'react-redux'** which is used to extract data from the Redux store state, using a selector function.
2. with this **useSelector** we accessed pincode data from store state which is assigned to **pincodeSelectorData** variable.

```
useEffect(() =>
    if (sheduleSelectorData.shedule.length == 0) {
       dispatch(getSheduleData);
     }

      if (sheduleSelectorData.shedule.length > 0) {
       for (; i < sheduleSelectorData.shedule.length; i += 1) {
         var singleSheduleData = sheduleSelectorData.shedule;
         if (singleSheduleData[i]._id == id) {
           setChecked(
            ...checked,
             singleSheduleData[i].cSheduleFrequency.split(",")
          );

           if (singleSheduleData[i].cSheduleType == "1") {
             setConsoleFlag(true);
             setB2bFlag(false);
          }

          if (singleSheduleData[i].cSheduleType == "2") {
             setConsoleFlag(false);
            setB2bFlag(true);
          }
          setSheduleData({
               cShedulepol: singleSheduleData[i].cShedulepol,
            cShedulepod: singleSheduleData[i].cShedulepod,
            cSheduleRouting: singleSheduleData[i].cSheduleRouting,
            cSheduleTransit: singleSheduleData[i].cSheduleTransit,
            cBookingCutoffFormat:singleSheduleData[i].cBookingCutoffFormat,
            cHandoverCutoffFormat:singleSheduleData[i].cHandoverCutoffFormat,
            cvalidUntil:singleSheduleData[i].cvalidUntil,
            cSheduleType: singleSheduleData[i].cSheduleType,
            cSheduleETA: singleSheduleData[i].cSheduleETA,
            cSheduleETD: singleSheduleData[i].cSheduleETD,
            cSheduleHeightLimit: singleSheduleData[i].cSheduleHeightLimit,
            cSStatus: singleSheduleData[i].cSStatus,
          });

           reset(singleSheduleData[i]);
        }
      }
  }

    if (originData.origin.length == 0) {
      dispatch(getOriginData());
     }
     if (destinationData.destination.length == 0) {
       dispatch(getDestinationData());
     }
  }, []);
```

3. in this useEffect we are using if condition with for loop.
4. **reset** the entire form state, fields reference, and subscriptions. There are optional arguments and will allow partial form state reset.
5. with help of for loop we are getting correct data with help of unique id and that data we are storing in **sheduleData** state.
6. we are checking that data id with params id , it helps use to get correct single data.
7. its helping us to retrive data in edit page.
8. **errors** is the userFrom field which contains error messeges.
9. **updateSheduleData** is function dispatched using dispatch function. which update the value and store in api.
10. The **useParams** hook returns an object of key/value pairs of the dynamic params from the current URL that were matched by the **Route path**
    . Child routes inherit all params from their parent routes.
11. **isValid** property return boolean value.
12. **setSheduleProcessData** this is onChage function. and The onchange event occurs when the value of an element has been changed. and in this function we are storing all input value in **sheduleData** state.

3. **List.js**

```
import { useEffect, useState } from "react";
import * as XLSX from "xlsx";
import Header from "../Common/header";
import Sidebar from "../Common/sidebar";
import { RiFolderUploadLine } from "react-icons/ri";
import "./index.css";
import { Link, useNavigate } from "react-router-dom";
import { useDispatch, useSelector } from "react-redux";
import {
  getSheduleData,
  deleteSheduleData,
  multipleDeleteSheduleData,
  multipleSaveSheduleData,
} from "../../redux/actions/sheduleAction";
import { getOriginData } from "../../redux/actions/originAction";
import { getDestinationData } from "../../redux/actions/destinationAction";
import consolFile from "../SampleExcelFile/Console.xls";
import backtobackFile from "../SampleExcelFile/backtoback.xls";
import { MdDownload } from "react-icons/md";

function Shedule() {
  const [excelFile, setExcelFile] = useState([]);
  const [excelData, setExcelData] = useState([]);
  const [filename, setFileName] = useState("");
  const [checked, setChecked] = useState(false);
  const createdBy = sessionStorage.getItem("dokoid");
  const dispatch = useDispatch();
  const navigate = useNavigate();
  var originData = useSelector((state) => state.originDatas);
  var destinationData = useSelector((state) => state.destinationDatas);
  var sheduleData = useSelector((state) => state.sheduleDatas);
  //console.log(hsns);

  const fileType = ["application/vnd.ms-excel"];
  const handleFile = (e) => {
    let selectedFile = e.target.files[0];

    setFileName(selectedFile.name);

    if (selectedFile && fileType.includes(selectedFile.type)) {
      let reader = new FileReader();
      reader.readAsArrayBuffer(selectedFile);
      reader.onload = (e) => {
        console.log(e);
        // debugger;
        setExcelFile(e.target.result);
        alert("File Uploaded");
      };
    } else {
      setExcelFile(null);
      console.log("plz select your file");
    }
  };

  console.log(sheduleData);

  function changeDestinationId(Destination) {
    var polArr = [];
    destinationData.destination.map((value, i) => {
      if (Destination === value.dName) {
        polArr.push(value._id);
        console.log(value._id);
      }
    });
    return polArr[0];
  }

  function changeOriginId(Origin) {
    var podArr = [];
    originData.origin.map((value, i) => {
      if (Origin === value.oName) {
        podArr.push(value._id);
        console.log(value._id);
      }
    });
    return podArr[0];
  }
  // submit function
  const handleSubmit = (e) => {
    e.preventDefault();

    if (excelFile !== null) {
      const workbook = XLSX.read(excelFile, { type: "buffer" });
      const worksheetName = workbook.SheetNames[0];
      const worksheet = workbook.Sheets[worksheetName];
      const data = XLSX.utils.sheet_to_json(worksheet, { raw: false });
      let finalExcelData = [];
      for (var i = 0; i < data.length; i += 1) {
        // debugger;
        let dataObject = Object.assign({}, data[i]);
        dataObject.cShedulepol = changeOriginId(data[i].cShedulepol);
        dataObject.cShedulepod = changeDestinationId(data[i].cShedulepod);

        console.log(dataObject);
        finalExcelData.push(dataObject);
        //debugger;
      }
      //console.log(finalExcelData);

      dispatch(multipleSaveSheduleData(finalExcelData));
      setFileName("");
      setExcelFile("");
      //dispatch(getLclSheduleData());
      window.location.reload();
    } else {
      setExcelData(null);
      alert("Upload Only xls file");
      setFileName("");
    }
    // setExcelData(data);
  };

  useEffect(() => {
    dispatch(getSheduleData());
    if (originData.origin.length === 0) {
      dispatch(getOriginData());
    }
    if (destinationData.destination.length === 0) {
      dispatch(getDestinationData());
    }
  }, []);

  const multicheck = document.getElementById("multicheck");
  const allDeleteHandle = (e) => {
    let checkboxSelectedIds = [];
    let checkboxSelected = document.querySelectorAll(
      ".sheduleCheckbox:checked"
    );
    checkboxSelected.forEach(function checkboxNode(node, index) {
      checkboxSelectedIds.push({ id: node.value, createdBy: createdBy });
    });

    if (checkboxSelectedIds.length === 0) {
      return false;
    }

    dispatch(multipleDeleteSheduleData(checkboxSelectedIds));
    removeCheck();
    multicheck.checked = false;
  };

  const checkFullData = (e) => {
    let checkboxSelected = document.querySelectorAll(".sheduleCheckbox");
    let isCheckboxChecked = e.target.checked;

    if (isCheckboxChecked === true) {
      checkboxSelected.forEach(function checkboxNode(node, index) {
        node.checked = true;
      });
    } else {
      checkboxSelected.forEach(function checkboxNode(node, index) {
        node.checked = false;
      });
    }
  };

  function removeCheck() {
    const checkbox = document.querySelectorAll(".sheduleCheckbox");

    console.log(checkbox);

    checkbox.forEach((check) => {
      check.checked = false;
    });
  }

  function deleteHandle(_id) {
    const uniqueId = [{ id: _id, createdBy: sessionStorage.getItem("dokoid") }];

    dispatch(deleteSheduleData(uniqueId));
    dispatch(getSheduleData());
    removeCheck();
  }

  return (
    <div className="main" id="main">
      <Header />
      <Sidebar />
      <div className="form">
        <div className="page-heading">
          <h4>Air Schedule List</h4>
        </div>
        <form className="form-group" onSubmit={handleSubmit}>
          <div className="upload-div">
            <div>
              <label htmlFor="uploadicon">
                <p className="upload-icon">
                  <RiFolderUploadLine />
                </p>
              </label>
              <input
                type="file"
                id="uploadicon"
                onChange={handleFile}
                required
              ></input>
            </div>
            <div>
              <p className="filename">{filename}</p>
            </div>
            <div>
              <button
                type="submit"
                className="submit-btn"
                style={{ marginTop: 5 + "px" }}
              >
                Submit
              </button>
            </div>
            <h5 className="danger">*Supported Files are xls only</h5>
          </div>
          <div className="sample-file">
            <a href={consolFile} download="console.xls">
              Console-Excel <MdDownload />
            </a>
            <br />
            <a href={backtobackFile} download="backtoback.xls">
              Back to Back-Excel <MdDownload />
            </a>
          </div>
        </form>
      </div>

      <div className="add-delete-btn">
        <div className="add">
          <button onClick={() => navigate("/shedule/add/")} className="add-btn">
            Add
          </button>
        </div>
        <div className="delete-div">
```

```
            <button
              className="delete-btn"
              data-bs-toggle="modal"
              data-bs-target="#alldeleteModal"
            >
              Delete
            </button>
          </div>
        </div>

        <div className="viewer">
          {sheduleData.shedule.length === 0 && <>No file selected</>}
          {sheduleData.shedule.length > 0 && (
            <div className="table-responsive">
              <table id="example" className="table">
                <thead>
                  <tr>
                    <th>
                      <input
                        type="checkbox"
                        onClick={checkFullData}
                        id="multicheck"
                      />
                    </th>
                    {/* <th>SCHEDULE ID</th> */}
                    <th>POL</th>
                    <th>POD</th>
                    <th>ROUTING</th>
                    <th>TRANSIT</th>
                    <th>FREQUENCY</th>
                    <th>BOOKING CUT OFF FORMAT</th>
                    <th>HANDOVER CUT OFF FORMAT</th>
                    <th>VALID UNTIL</th>
                    {/* <th>BOOKING CUTOFF</th>
                    <th>HANDOVER CUTOFF</th>
                    <th>COOLING PERIOD</th> */}
                    <th>HEIGHT LIMIT</th>
                    <th>SCHEDULE TYPE</th>
                    <th>ACTION</th>
                  </tr>
                </thead>
                <tbody>
                  {sheduleData.shedule.map((e, index) => {
                    let sType = "",
                      spol = [],
                      spod = [];

                    if (e.cSheduleType === "1") {
                      sType = "console";
                    } else {
                      sType = "back-2-back";
                    }

                    if (originData.origin.length > 0) {
                      let spolArray = originData.origin.filter((value, index) => {
                        if (e.cShedulepol === value._id) {
                          return e.cShedulepol === value._id;
                        } else if (e.cShedulepol.toLowerCase() === value.oName) {
                          return e.cShedulepol === value.oName;
                        }
                      });
                      if (spolArray.length > 0) {
                        spol = spolArray[0].oName;
                      }
                    }

                    if (destinationData.destination.length > 0) {
```

```
          let spodArray = destinationData.destination.filter(
            (value, index) => {
              if (e.cShedulepod.toLowerCase() === value.dName) {
                return e.cShedulepod === value.dName;
              } else if (e.cShedulepod === value._id) {
                return e.cShedulepod === value._id;
              }
            }
          );
          //spod = spodArray[0].dName;
          if (spodArray.length > 0) {
            spod = spodArray[0].dName;
          }
        }

        return (
          <tr key={index}>
            <td>
              <input
                type="checkbox"
                className="sheduleCheckbox"
                value={e._id}
              />
            </td>
            {/* <td>{e.cSheduleID}</td> */}
            <td>{spol}</td>
            <td>{spod}</td>
            <td>{e.cSheduleRouting}</td>
            <td>{e.cSheduleTransit}</td>
            <td>{e.cSheduleFrequency}</td>
            <td>{e.cBookingCutoffFormat}</td>
            <td>{e.cHandoverCutoffFormat}</td>
            <td>{e.cvalidUntil}</td>
            {/* <td>{e.cSheduleBookingCutoff}</td>
            <td>{e.cShedulecargoHandoverCutoff}</td>
        <td>{e.cSheduleCoolingPeriod}</td> */}
            <td>{e.cSheduleHeightLimit}</td>
            <td>{sType}</td>

            <td>
              <div className="action-div">
                <div>
                  <button
                    onClick={() => navigate("/shedule/edit/" + e._id)}
                    className="add-btn"
                  >
                    Edit
                  </button>
                  <button
                    className="delete-btn"
                    onClick={() => deleteHandle(e._id)}
                  >
                    Delete
                  </button>
                </div>
                <>
                  {/* Button trigger modal */}
                  {/* Modal */}
                  {/* <div
                    className="modal fade"
                    id="exampleModal"
                    tabIndex={-1}
                    aria-labelledby="exampleModalLabel"
                    aria-hidden="true"
                  >
                    <div className="modal-dialog">
                      <div className="modal-content">
                        <div className="modal-header">
                          <h1
                            className="modal-title fs-5"
                            id="exampleModalLabel"
                          >
                            Schedule Delete
                          </h1>
                          <button
                            type="button"
                            className="btn-close"
                            data-bs-dismiss="modal"
                            aria-label="Close"
                          />
                        </div>
                        <div className="modal-body">
                          Are you want to Delete?
                        </div>
                        <div className="modal-footer">
                          <button
                            type="button"
                            className="btn btn-secondary"
                            data-bs-dismiss="modal"
                          >
                            Cancel
                          </button>
                          <button
                            type="button"
                            className="btn btn-primary"
                            data-bs-dismiss="modal"
                            onClick={() => deleteHandle(e._id)}
                          >
                            Delete
                          </button>
                        </div>
                      </div>
                    </div>
                  </div> */}
                </>
              </div>
            </td>
          </tr>
        );
      })}
    </tbody>
  </table>
  <>
    {/* Modal */}
    <div
      className="modal fade"
      id="alldeleteModal"
      tabIndex={-1}
      aria-labelledby="exampleModalLabel"
      aria-hidden="true"
    >
      <div className="modal-dialog">
        <div className="modal-content">
          <div className="modal-header">
            <h1
              className="modal-title fs-5"
              id="exampleModalLabel"
            ></h1>
            <button
              type="button"
              className="btn-close"
              data-bs-dismiss="modal"
              aria-label="Close"
            />
```

```
                    </div>
                    <div className="modal-body">
                      Are you want to Delete All?
                    </div>
                    <div className="modal-footer">
                      <button
                        type="button"
                        className="btn btn-secondary"
                        data-bs-dismiss="modal"
                      >
                        Cancel
                      </button>
                      <button
                        type="button"
                        data-bs-dismiss="modal"
                        className="btn btn-primary"
                        onClick={allDeleteHandle}
                      >
                        Delete
                      </button>
                    </div>
                  </div>
                </div>
              </div>
            </>
          </div>
        )}
      </div>
    </div>
  );
}

export default Shedule;
```

▼ **Description**

1. **useSelector** is imported from **'react-redux'** which is used to extract data from the Redux store state, using a selector function.
2. **useNavigate** is a return function , with this help we can navigate from one path URL to another path URL.

```
function removeCheck() {
    const checkbox = document.querySelectorAll(".sheduleCheckbox");
    console.log(checkbox);
    checkbox.forEach((check) => {
      check.checked = false;
      console.log(check);
    });
  }
```

3. **removeCheck** function is used to chekced checkboxes is convert into unchecked chekboxes.
4. with **querySelectorAll** we are selecting all the input checkboxes whose have **rateCheckbox** className.

5. The **forEach** method calls a function for each element in an array . it is not executed for empty elements.
6. with help of **forEach** method we are making condtion false to all checkboxes. so that we are removing checked of checkboxes.

```
<a href={sheduleFile} download="shedule.xls">
            Sample-Excel <MdDownload />
          </a>
```

7. This is **<a>** tag with download attribute is used to download the given file from browser.

```
const allDeleteHandle = (e) => {
  let checkboxSelectedIds = [];
  let checkboxSelected = document.querySelectorAll(
    ".sheduleCheckbox:checked"

  checkboxSelected.forEach(function checkboxNode(node, index) {
    checkboxSelectedIds.push({ id: node.value, createdBy: createdBy });
  });

  if (checkboxSelectedIds.length === 0) {
    return false;
  }
  dispatch(multipleDeleteSheduleData(checkboxSelectedIds));
  removeCheck();
  multicheck.checked = false;
};
```

6. this function is used to delete all the selected rows from table.
7. **dispatch** is used to dispatch the **multipleDeleteSheduleData** function.
8. **checkboxSelectedIds** in this array we are pushing the id as the checkbox value and createdBy as object id.
9. ‘’ **sheduleCheckbox:checked ‘’** with this help we are getting only checked checkboxes.
10. **removeCheck** this function is called here to remove checked tick of checkboxes.
11. **push** method is used to adds new items **to the end**
 of an array.
12. in this component we used some react-icons also.

```
const checkFullData = (e) => {
  let checkboxSelected = document.querySelectorAll(".sheduleCheckbox");
  let isCheckboxChecked = e.target.checked;

  if (isCheckboxChecked === true) {
    checkboxSelected.forEach(function checkboxNode(node, index) {
      node.checked = true;
    });
  } else {
    checkboxSelected.forEach(function checkboxNode(node, index) {
      node.checked = false;
    });
  }
};
```

9. **checkFullData** function is used check tick of checkboxes what we want to select.
10. The **querySelectorAll** method returns all elements that matches a CSS selector.

```
  function deleteHandle(_id) {
    const uniqueId = [{ id: _id, createdBy: sessionStorage.getItem("dokoid") }];

    dispatch(deleteSheduleData(uniqueId));
    dispatch(getSheduleData());
    removeCheck();
  }
```

- **deleteHandle** this function is used to single delete pincode.
- **uniqueId** is the array of object contains unique id .
- it dispatch the **deleteSheduleData** action.

10. **handleSubmit function**

```
const handleSubmit = (e) => {
    e.preventDefault();

    if (excelFile !== null) {
      const workbook = XLSX.read(excelFile, { type: "buffer" });
      const worksheetName = workbook.SheetNames[0];
      const worksheet = workbook.Sheets[worksheetName];
      const data = XLSX.utils.sheet_to_json(worksheet);
      // setExcelData(data);
      console.log(data);
      // debugger;
      dispatch(multipleSaveSheduleData(data));
      // window.location.reload();
    } else {
      setExcelData(null);
      alert("Upload Only xls file");
      setFilename("");
    }
  };
```

- **handleSubmit** function is used to get all the data from uploaded sheet and it converted into json format and this dat a passed to this **multipleSaveSheduleData** function.
- **XLSX** is the NPM package used here to read the xls file and that file data converted into json format.
- **sheet_to_json** create array of json value from worksheet.
- **XLSX.read** is used to read the excel file.
- **excelFile** this state variable contains uploaded excel file.

1. **handleFile** Function

```
const fileType = ["application/vnd.ms-excel"];
const handleFile = (e) => {
  let selectedFile = e.target.files[0];
  console.log(selectedFile.name);
  setFilename(selectedFile.name);

  if (selectedFile && fileType.includes(selectedFile.type)) {
    let reader = new FileReader();
    reader.readAsArrayBuffer(selectedFile);
    reader.onload = (e) => {
      console.log(e);
      // debugger;
      setExcelFile(e.target.result);
      alert("File Uploaded");
    };
  } else {
    setExcelFile(null);
    console.log("plz select your file");
    alert("Upload Only xls file");
  }
};
```

- **handleFile** function is the onchange function used to get uploaded file in input file type.
- **e.target.files[0]** with this we are getting uploaded file.
- here we are checking file type to xls .
- The **includes**() method returns **true** if a string contains a specified string.
- **FileReader** it helps web browser to read asynchronously content of file.
- **readAsArrayBuffer** method is **used to start reading the contents of a specified Blob or File.**

2. var **sheduleData** = useSelector((state) => state.sheduleDatas);

- **sheduleData** contains array of object .R
- with this variable we are mapping data in table format.
- **map** function creates a new array from calling a function for every array element.
  - it calls the function once for each element of array
  - it does not change the original element.

3. The useNavigate hook returns a function that helps us to navigate programmatically,
4. A <**Link**> is an element that helps us navigate to another page by clicking on it.

Thank You 🙂…!