import "./StudentDetailsFormCss.css";

import { AdapterDayjs } from "@mui/x-date-pickers/AdapterDayjs";

import { Box, MenuItem, Step, StepLabel, Stepper } from "@mui/material";

import { DatePicker } from "@mui/x-date-pickers/DatePicker";

import { FooterStyle } from "../../StyleCss/FooterStyle";

import { LocalizationProvider } from "@mui/x-date-pickers";

import { studentDTO, Students1 } from "../../../Types/ModelsStudents";

import { useNavigate } from "react-router-dom";

import { useState } from "react";

import DeleteForeverOutlinedIcon from "@mui/icons-material/DeleteForeverOutlined";

import Select from "react-select";

import TextField from "@mui/material/TextField";

import { statuses, armyUnits, homeStatus, statusesAddStudent } from "../constants";

import WifeDetails from "../UpdateStudent/WifeDetails";

const StudentDetailsForm = ({

  myStudent,

  onNextClick,

  setMyStudent,

  setShowFilesForm,

  activeStep,

  setActiveStep,

}: {

  myStudent: studentDTO;

  onNextClick: () => void;

  setMyStudent: (student: studentDTO) => void;

  setShowFilesForm: any;

  activeStep: any;

  setActiveStep: any;

}) => {

  const navigate = useNavigate();

  const [errors, setErrors] = useState<any>({});

  const [currentStep, setCurrentStep] = useState<number>(0);

  const steps = ["", ""];

  const radioHandler1 = (event: any) => {

    if (event.target.value === "yes")

      return setMyStudent({ ...myStudent, wasInSeminar: true });

    setMyStudent({ ...myStudent, wasInSeminar: false });

  };

  const radioHandler = (event: any) => {

    if (event.target.value === "yes")

      return setMyStudent({ ...myStudent, wantChavruta: true });

    setMyStudent({ ...myStudent, wantChavruta: false });

  };

  const refreshPage = () => {

    window.location.reload();

  };

  const validateName = () => {

    if (!myStudent.firstName) {

      return setErrors({ ...errors, firstName: "First name is required." });

    }

    if (myStudent.firstName.length === 1)

      return setErrors({ ...errors, firstName: "First name is invalid." });

    if (/^\d+$/.test(myStudent.firstName))

      return setErrors({ ...errors, firstName: "First name is invalid." });

    setErrors({ ...errors, firstName: "" });

  };

  const validateId = () => {

    if (!myStudent.studentId) {

      return setErrors({ ...errors, studentId: "ID number is required." });

    }

    if (

      myStudent.studentId.length !== 9 ||

      !/^\d+$/.test(myStudent.studentId)

    ) {

      return setErrors({ ...errors, studentId: "ID Number is invalid." });

    }

    setErrors({ ...errors, studentId: "" });

  };

  const changeArmiUnit = (event: any) => {

    let value = Array.from(event.label, option => option).toString();

    console.log("value");

    console.log(event.label);

    setMyStudent({ ...myStudent, armiUnit: value });

    // setMyStudent({ ...myStudent, armiUnit: event.label });

  };

  const changeStatus = (event: any) => {

    setMyStudent({ ...myStudent, statusId: parseInt(event.label.value) });

  };

  const changeHomeStatus = (event: any) => {

    setMyStudent({ ...myStudent, homeStatus: event.label });

  };

  const validEmail = () => {

    if (!myStudent.email) {

      return setErrors({ ...errors, email: "Email is required." });

    }

    if (

      !/^((?!\.)[\w-\_.]\*[^.])(@\w+)(\.\w+(\.\w+)?[^.\W])$/gim.test(

        myStudent.email

      )

    )

      return setErrors({ ...errors, email: "Email number is invalid." });

    setErrors({ ...errors, email: "" });

  };

  const prev = (step: any) => {

    setShowFilesForm(step === 2);

  };

  return (

    <div id="mainDiv">

      <Box id="stepper" sx={{ width: "15%" }}>

        <Stepper activeStep={activeStep} alternativeLabel>

          {steps.map((label, index) => (

            <Step key={index}>

              <StepLabel onClick={() => setActiveStep(index)}>

                {label}

              </StepLabel>

            </Step>

          ))}

        </Stepper>

      </Box>

      <h1 id="titleDetils">פרטים אישיים</h1>

      <TextField

        // error={true}

        // helperText={"avh"}

        className="firstName"

        id="outlined-basic"

        label="שם פרטי"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, firstName: e.target.value })

        }

        value={myStudent.firstName}

        onBlur={validateName}

        error={!!errors.firstName}

        helperText={errors.firstName}

      ></TextField>

      <TextField

        className="lastName"

        id="outlined-basic"

        label="שם משפחה"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, lastName: e.target.value })

        }

        value={myStudent.lastName}

        onBlur={validateName}

        error={!!errors.lastName}

        helperText={errors.lastName}

      />

      <TextField

        className="id"

        id="outlined-basic"

        label="תעודת זהות"

        variant="outlined"

        type={"number"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, studentId: e.target.value })

        }

        value={myStudent.studentId}

        onBlur={validateId}

        error={!!errors.studentId}

        helperText={errors.studentId}

      />

      <input

        value={myStudent.birthdate.toISOString().split('T')[0]}

        className="birthdateStudent"

        type={"date"}

        onChange={(e: any) =>

          setMyStudent({ ...myStudent, birthdate: new Date(e.target.value) })

        }

      ></input>

      <TextField

        className="fatherName"

        id="outlined-basic"

        label="שם האב"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, fatherName: e.target.value })

        }

        value={myStudent.fatherName}

      />

      <TextField

        className="motherName"

        id="outlined-basic"

        label="שם האם"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, motherName: e.target.value })

        }

        value={myStudent.motherName}

      />

      <TextField

        className="proffesion"

        id="outlined-basic"

        label="מקצוע / התמחות"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, profession: e.target.value })

        }

        value={myStudent.profession}

      />

      <br></br>

      <Box

        className="status"

        component="form"

        sx={{

          "& .MuiTextField-root": { m: 1, width: "25ch" },

        }}

        noValidate

        autoComplete="off"

      >

        <TextField

          id="outlined-select-currency"

          select

          label="סטטוס"

          defaultValue="1"

          // helperText="סמן מהו הסטטוס שלך"

          onChange={(e) =>

            setMyStudent({ ...myStudent, statusId: parseInt(e.target.value) })

          }

          value={myStudent.statusId}

        >

          {statusesAddStudent.map((option) => (

            <MenuItem key={option.value} value={option.value}>

              {option.label}

            </MenuItem>

          ))}

        </TextField>

        {myStudent.statusId === 3 ? (

          <WifeDetails

            myStudent={myStudent}

            setMyStudent={setMyStudent}

          ></WifeDetails>

        ) : (

          <></>

        )}

      </Box>

      <TextField

        className="school"

        id="outlined-basic"

        label="תיכון"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) => setMyStudent({ ...myStudent, school: e.target.value })}

        value={myStudent.school}

      />

      <br></br>

      <Box

        className="armiUnit"

        component="form"

        sx={{

          "& .MuiTextField-root": { m: 1, width: "25ch" },

        }}

        noValidate

        autoComplete="on"

      >

        <div>

          <div className="containerArmi ">

            <div>

              <Select

                className="containerArmi "

                // value={myStudent.armiUnit}

                options={armyUnits}

                onChange={changeArmiUnit}

                placeholder="יחידת שירות בצבא"

                isRtl

              />

            </div>

          </div>

        </div>

      </Box>

      <TextField

        className="phone"

        id="outlined-basic"

        label="פלאפון"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) => setMyStudent({ ...myStudent, phone: e.target.value })}

        value={myStudent.phone}

      />

      <TextField

        className="mail"

        id="outlined-basic"

        label="מייל"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) => setMyStudent({ ...myStudent, email: e.target.value })}

        onBlur={validEmail}

        error={!!errors.email}

        helperText={errors.email}

        value={myStudent.email}

      />

      {/\* <TextField className="armiUnit"id="outlined-basic" label="יחידה בצבא" variant="outlined" type={"string"} required onChange={(e) => setMyStudent({ ...myStudent, armiUnit: e.target.value})} /><br></br> \*/}

      <TextField

        className="adress"

        id="outlined-basic"

        label="כתובת"

        variant="outlined"

        type={"string"}

        required

        onChange={(e) => setMyStudent({ ...myStudent, adress: e.target.value })}

        value={myStudent.adress}

      />

      <br></br>

      <TextField

        className="zipCode"

        id="outlined-basic"

        label="מיקוד"

        variant="outlined"

        type={"number"}

        required

        onChange={(e) =>

          setMyStudent({ ...myStudent, zipCode: parseInt(e.target.value) })

        }

        value={myStudent.zipCode}

      />

      <br></br>

      <br></br>

      <Box

        className="homeStatus"

        component="form"

        sx={{

          "& .MuiTextField-root": { m: 1, width: "25ch" },

        }}

        noValidate

        autoComplete="on"

      >

        <div>

          <div className="homeStatus ">

            <div>

              <Select

                className="homeStatus "

                options={homeStatus}

                onChange={changeHomeStatus}

                placeholder="מצב בית"

                isRtl

              />

            </div>

          </div>

        </div>

      </Box>

      <br></br>

      <legend className="container0">?מעוניין בחברותא</legend>

      <div className="container" onChange={radioHandler}>

        <input

          type="radio"

          name="wantChavruta"

          value="yes"

          checked={myStudent.wantChavruta === true}

          readOnly

        />

        <label htmlFor="true">כן</label>

        <input

          type="radio"

          name="wantChavruta"

          value="false"

          checked={myStudent.wantChavruta === false}

          readOnly

        />

        <label htmlFor="false">לא</label>

      </div>

      <legend className="container11">?השתתף בסמינר</legend>

      <div className="container1" onChange={radioHandler1}>

        <input

          type="radio"

          name="wasInSeminar"

          value="yes"

          checked={myStudent.wasInSeminar === true}

          readOnly

        />

        <label htmlFor="true">כן</label>

        <input

          type="radio"

          name="wasInSeminar"

          value="false"

          checked={myStudent.wasInSeminar === false}

          readOnly

        />

        <label htmlFor="false">לא</label>

      </div>

      {<FooterStyle></FooterStyle>}

      <h1 id="titleDetils1">פרטי התקשרות</h1>

      <h1 id="titleDetils2">כתובת מגורים</h1>

      {/\* <h1 id="titleDetils3">פרטי האישה </h1> \*/}

      <button

        id="nextButton"

        onClick={() => {

          const hasErrors = Object.values(errors).some((e) => !!e);

          hasErrors && alert("הנתונים לא חוקיים");

          !hasErrors && setActiveStep(1);

        }}

        // disabled={}

      >

        הבא

      </button>

      <button id="garbageButton" onClick={refreshPage}>

        {

          <DeleteForeverOutlinedIcon

            style={{ width: "35px", height: "35px", color: "white" }}

          />

        }

      </button>

    </div>

  );

};

export default StudentDetailsForm;