# SEMINAR REPORT

on

# Website for educational institutions

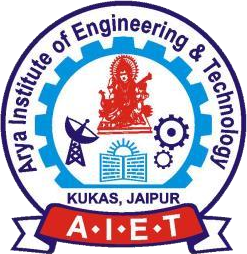
*submitted in partial fulfillment*

*for the award of the Degree of*

***Bachelor of Technology***

### in

**Computer Science & Engineering**



|  |  |
| --- | --- |
| **Submitted To:** | **Submitted By:** |
| Dr. Manish Kumar Mukhija | Kusum Gautam |

Associate Professor & HOD 19EAYCS058

### Department of Computer Science & Engineering Arya Institute of Engineering & Technology, Jaipur

**Rajasthan Technical University, Kota (2022-23)**

***Candidate’s Declaration***

I hereby declare that the work, which is being presented in the Industrial Training report, entitled **“React Native”** in partial fulfilment for the award of Degree of “Bachelor of Technology” in Department of Computer Science & Engineering with Specialization in Computer Engineering and submitted to the **Department of Computer Science & Engineering,** Arya Institute of Engineering & Technology, under the Guidance of Dr. Manish Kumar Mukhija, Associate Professor & HOD and Mr. Satish Kumar Alaria, Assistant Professor, Department of Computer Science & Engineering.

**(Kusum Gautam)**

B. Tech. Student, Department of Computer Science & Engineering Roll No.: 19EAYCS058

Arya Institute of Engineering & Technology, Jaipur

**(Dr. Manish Kumar Mukhija)**

Associate Professor & HOD, Department of Computer Science & Engineering Arya Institute of Engineering & Technology, Jaipur

### Acknowledgement

“There are times when silence speaks so much more loudly than words of praise to only as good as be little person, whose words do not express, but only put a veneer over true feelings, which are of gratitude at this point of time.”

“Presentation, inspiration and motivation have always played a key role in the success of any venture.”

I pay my deep sense of gratitude to Mr. Manish Mukhija, Head of Department, Arya institute of engineering and technology, Kukas, Jaipur to encourage me to the highest peak and to provide me the opportunity to prepare the project. I am immensely obliged to my friends for their elevating inspiration, encouraging guidance and kind supervision in the completion of my project.

I feel to acknowledge my indebtedness a deep sense of gratitude to my guide Mr. Satish Alaria, Training Coordinator, Arya institute of engineering and technology.

Last, but not the least, my parents are also an important inspiration for me and also like to thank all those who directly or indirectly helped me throughout my work. So, with due regards, I express my gratitude to them.

**Kusum Gautam**

## TABLE OF CONTENTS

|  |  |
| --- | --- |
| **Candidate’s Declaration** | |
| **Acknowledgment** | |
| **Certificate** |  |
| **Table of Contents** | |
| **Abstract** |  |
| **Chapter 1:** | **INTRODUCTION**   * **What is this website about?** * **What is the primary language used?** * **What is JavaScript?** * **What is MERN stack?** |
| **Chapter 2:**  **Chapter 3:**  **Chapter4:**  **Chapter 5:**  **Chapter 6:**  **Chapter 7:**  **Chapter 8:**  **CONCLUSION** | **ADVANTAGES OF MERN STACK TECHNICAL DETAILS**  **USER SET**  **CODE**  **IMPLEMENTATION**  **IMAGES SECURITY** |

Chapter 1

### Introduction

* a website for an educational institution is like its face. so, without a website, it is like a person with a covered face. it is the website that helps in unveiling the educational institutions in front of all the prospects.
* Having a website for your educational institution can help you to show your institution’s infrastructure, quality of education, geographic location, and other specific features of your institution. a good website for educational institutions could attract more aspirants. this website is all about this, however, it contains certain additional features which other websites don’t provide. It provides you filters, saving your searched output, comparing the different institutions and many more such features that other institutions do not provide.
* this website is established on MERN Stack. As whenever websites are created using MERN technologies, then it is always SEO friendly. Better performance: Better performance refers to the faster response between the backend and front-end and database, which ultimately improves the website speed and yields better performance, thus providing a smooth user experience.
* **THE PRIMARY LANGUAGE USED IN IT:**

The primary language used in this website besides other tools and technologies used is JavaScript.

* **WHAT IS JavaScript**:

Javascript is used by programmers across the world to create dynamic and interactive web content like applications and browsers. JavaScript is so popular that it's the most used programming language in the world, used as a client-side programming language by 97.0% of all websites. Javascript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, two of which ([HTML](https://developer.mozilla.org/en-US/docs/Learn/HTML) and [CSS](https://developer.mozilla.org/en-US/docs/Learn/CSS)). For the scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies. In the backend, there is a JavaScript stack as well. It is a MEAN stack, which includes MongoDB, ExpressJS, AngularJS, and NodeJS.

* **WHAT IS MERN STACK?**

MERN stands for MongoDB, Express, React, and Node, after the four key technologies that make up the stack.

* MongoDB — document database
* Express(.js) — Node.js web framework
* React(.js) — a client-side JavaScript framework
* Node(.js) — the premier JavaScript web server

Express and Node make up the middle (application) tier. Express.js is a server-side web framework, and Node.js is a popular and powerful JavaScript server platform. Regardless of which variant you choose, ME(RVA)N is the ideal approach to working with JavaScript and JSON, all the way through.  MERN is a full stack, following the traditional three-tier architectural pattern, including the front-end display tier (React.js), application tier (Express.js and Node.js), and database tier (MongoDB).

CHAPTER2

* **ADVANTAGES OF USING MERN STACK**
* MERN is the abbreviation for MongoDB, Express.JS, ReactJS, and NodeJS. It is an open-source framework utilized in developing dynamic websites and mobile apps.
* A fast and resilient web server saves time spent on troubleshooting input/output and CPU issues.
* It provides support to both front-end andback-end app development.
* More responsive to user inputs, offering a better user experience

CHAPTER3

**TECHNICAL DETAILS**

* this section includes the front-end and back-end technologies used to build this website**.**

## FRONTEND

## HTML, CSS, JS, REACT JS

* + BACKEND
  + PHP, APIs
  + MongoDB
  + EXPRESSJS
  + NODEJS

CHAPTER4

**USER SET**

# THERE ARE TWO USER SETS OF THE WEBSITE:

* + the one who sees and uses information about institutions
  + the one who uploads the information about the institution.

**CODE**

import React from "react";

import { BrowserRouter, Route, Routes } from "react-router-dom";

import {

  Home,

  About,

  SchoolDetails,

  Schools,

  Contace,

  ProfileMain,

} from "../screens";

import Footer from "../components/footer/Footer";

import Header from "../components/header/Header";

const AuthRoute = () => {

  return (

    <React.Fragment>

      <Header />

      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="about" element={<About />} />

        <Route path="schooldetails" element={<SchoolDetails />} />

        <Route path="schools" element={<Schools />} />

        <Route path="contact" element={<Contace />} />

        <Route path="profile" element={<ProfileMain />} />

      </Routes>

      <Footer />

    </React.Fragment>

  );

};

export default AuthRoute;

export const HEADERMENU\_PATH = {

  login: "/login",

  profile: "/profile",

  home: "/home",

};

import React from "react";

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

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

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

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

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

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

import Visibility from "@mui/icons-material/Visibility";

import VisibilityOff from "@mui/icons-material/VisibilityOff";

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

// import Item from "../../../utils/Items";

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

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

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

import { Container, Typography } from "@mui/material";

import { createTheme, styled } from "@mui/material/styles";

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

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

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

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

import GoogleIcon from "@mui/icons-material/Google";

import FacebookIcon from "@mui/icons-material/Facebook";

import { Formik } from "formik";

import \* as yup from "yup";

import { red } from "@mui/material/colors";

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

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

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

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

import { useState, useEffect } from "react";

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

import { ToastContainer, toast } from "react-toastify";

import axios from "axios";

import { useDispatch, useSelector } from "react-redux";

import { USER\_LOGGED\_IN } from "../../../redux/constants";

import BASE\_URL from "../../../utils/baseUrl";

import { GoogleLogin } from "react-google-login";

// import FacebookLogin from "react-facebook-login";

import FacebookLogin from "react-facebook-login/dist/facebook-login-render-props";

const appId = "5760285217358093";

const Item = styled(Paper)(({ theme }) => ({

  ...theme.typography.body2,

  textAlign: "center",

  color: theme.palette.text.secondary,

  height: 60,

  lineHeight: "60px",

}));

const ColorButton = styled(Button)(({ theme }) => ({

  color: theme.palette.getContrastText(red[500]),

  backgroundColor: red[500],

  "&:hover": {

    backgroundColor: red[700],

  },

}));

let LoginSchema = yup.object().shape({

  email: yup

    .string()

    .email("please enter a valid email")

    .required("email is required"),

  password: yup.string().required(),

});

const Login = () => {

  const dispatch = useDispatch();

  const dataTa = useSelector((state) => state);

  console.log(dataTa);

  const [showPass, setShowPass] = useState(false);

  const [showbackDrop, setShowbackDrop] = useState(false);

  const handleClickShowPassword = () => {

    setShowPass(!showPass);

  };

  const handleMouseDownPassword = (event) => {

    event.preventDefault();

  };

  const loginUser = async (values) => {

    setShowbackDrop(true);

    const getRes = await axios.post(BASE\_URL + "/api/login", values);

    try {

      const data = getRes;

      console.log(data.data);

      if (data.data.err == "Invalid Credientials") {

        setShowbackDrop(false);

        toast.error(data.data.err);

      } else if (data.data.mag == "user Login SucessFully.") {

        setShowbackDrop(false);

        toast.success("Logged in successfully");

        dispatch({ type: USER\_LOGGED\_IN, payload: data.data });

      } else if (data.data.err == "Invalid Credientials Pass") {

        setShowbackDrop(false);

        toast.error("Invalid Credentials");

      }

    } catch (e) {

      console.log(e);

    }

  };

  const responseGoogle = async (response) => {

    setShowbackDrop(true);

    const res = await response;

    try {

      setShowbackDrop(false);

      console.log(res);

    } catch (e) {

      console.log(e);

    }

  };

  const responseFacebook = (response) => {

    console.log(response);

  };

  return (

    <React.Fragment>

      <CssBaseline />

      <Container maxWidth={"sm"}>

        <Box

          maxWidth={"sm"}

          mx="auto"

          mt={5}

          //   border={1}

          style={{

            backgroundColor: "white",

            padding: 30,

            boxShadow: 1,

            borderRadius: 7,

          }}

          boxShadow={7}

        >

          <Box

            mb={4}

            display="flex"

            justifyContent={"center"}

            alignItems="center"

          >

            <Typography variant="h5">Login</Typography>

          </Box>

          <Formik

            initialValues={{ email: "", password: "" }}

            validationSchema={LoginSchema}

            validateOnMount={true}

            onSubmit={(values, { setSubmitting }) => {

              loginUser(values);

            }}

          >

            {({

              values,

              errors,

              touched,

              handleChange,

              handleBlur,

              handleSubmit,

              isSubmitting,

              isValid,

              /\* and other goodies \*/

            }) => (

              <form onSubmit={handleSubmit}>

                <Stack spacing={2}>

                  <TextField

                    id="outlined-basic"

                    label="Email"

                    variant="outlined"

                    name="email"

                    fullWidth

                    onChange={handleChange}

                    onBlur={handleBlur}

                    value={values.email}

                  />

                  <Typography variant="p" color={"red"} style={{ margin: 0 }}>

                    {errors.email && touched.email && errors.email}

                  </Typography>

                  <FormControl sx={{ m: 1 }} variant="outlined">

                    <InputLabel htmlFor="outlined-adornment-password">

                      Password

                    </InputLabel>

                    <OutlinedInput

                      id="outlined-adornment-password"

                      type={showPass ? "text" : "password"}

                      value={values.password}

                      onChange={handleChange}

                      onBlur={handleBlur}

                      name="password"

                      fullWidth

                      endAdornment={

                        <InputAdornment position="end">

                          <IconButton

                            aria-label="toggle password visibility"

                            onClick={handleClickShowPassword}

                            onMouseDown={handleMouseDownPassword}

                            edge="end"

                          >

                            {showPass == false ? (

                              <VisibilityOff />

                            ) : (

                              <Visibility />

                            )}

                          </IconButton>

                        </InputAdornment>

                      }

                      label="Password"

                    />

                  </FormControl>

                  <Typography variant="p" color={"red"} style={{ margin: 0 }}>

                    {errors.password && touched.password && errors.password}

                  </Typography>

                  <Stack

                    direction={"row"}

                    alignItems={"center"}

                    justifyContent={"space-between"}

                  >

                    <FormGroup>

                      <FormControlLabel

                        control={<Checkbox defaultChecked />}

                        label="Remember me"

                      />

                    </FormGroup>

                    <Typography>Forget password</Typography>

                  </Stack>

                  <Button variant="contained" type="submit" disabled={!isValid}>

                    Login

                  </Button>

                </Stack>

              </form>

            )}

          </Formik>

          <Stack>

            <GoogleLogin

              clientId="1051854074445-ftvvgr3kml6oqv7ujt9902qlbluvrve1.apps.googleusercontent.com"

              buttonText="Login with google"

              onSuccess={responseGoogle}

              render={(renderProps) => (

                <ColorButton

                  variant="contained"

                  startIcon={<GoogleIcon />}

                  style={{ marginTop: 10 }}

                  onClick={renderProps.onClick}

                  disabled={renderProps.disabled}

                >

                  Login with Google

                </ColorButton>

              )}

              onFailure={responseGoogle}

              cookiePolicy={"single\_host\_origin"}

            />

            <FacebookLogin

              appId="5760285217358093"

              autoLoad={false}

              fields="name,email,picture"

              callback={responseFacebook}

              icon="fa-facebook"

              render={(renderProps) => {

                return (

                  <Button

                    variant="contained"

                    startIcon={<FacebookIcon />}

                    style={{ marginTop: 10 }}

                    onClick={renderProps.onClick}

                  >

                    Login with Facebook

                  </Button>

                );

              }}

            />

          </Stack>

          <Stack

            style={{

              display: "flex",

              justifyContent: "center",

              alignItems: "center",

              marginTop: 10,

            }}

          >

            <Link

              to={"/register"}

              style={{ textDecoration: "none", color: "black" }}

            >

              <Typography>

                Already have account?

                <span style={{ color: "blueviolet" }}>Register</span>

              </Typography>

            </Link>

          </Stack>

        </Box>

      </Container>

      <Backdrop

        sx={{ color: "#fff", zIndex: (theme) => theme.zIndex.drawer + 1 }}

        open={showbackDrop}

        // onClick={handleClose}

      >

        <CircularProgress color="inherit" />

      </Backdrop>

    </React.Fragment>

  );

};

export default Login;

// import \* as React from "react";

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

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

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

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

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

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

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

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

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

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

import LockOutlinedIcon from "@mui/icons-material/LockOutlined";

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

// =====================

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

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

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

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

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

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

import Visibility from "@mui/icons-material/Visibility";

import VisibilityOff from "@mui/icons-material/VisibilityOff";

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

// import Item from "../../../utils/Items";

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

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

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

import { Container, Typography } from "@mui/material";

import { createTheme, styled } from "@mui/material/styles";

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

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

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

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

import GoogleIcon from "@mui/icons-material/Google";

import FacebookIcon from "@mui/icons-material/Facebook";

import { Formik } from "formik";

import \* as yup from "yup";

import { red } from "@mui/material/colors";

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

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

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

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

import { useState, useEffect } from "react";

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

import { ToastContainer, toast } from "react-toastify";

import axios from "axios";

import { useDispatch, useSelector } from "react-redux";

import { USER\_LOGGED\_IN } from "../../../redux/constants";

import BASE\_URL from "../../../utils/baseUrl";

import { GoogleLogin } from "react-google-login";

// import FacebookLogin from "react-facebook-login";

import FacebookLogin from "react-facebook-login/dist/facebook-login-render-props";

const appId = "5760285217358093";

const Item = styled(Paper)(({ theme }) => ({

  ...theme.typography.body2,

  textAlign: "center",

  color: theme.palette.text.secondary,

  height: 60,

  lineHeight: "60px",

}));

const ColorButton = styled(Button)(({ theme }) => ({

  color: theme.palette.getContrastText(red[500]),

  backgroundColor: red[500],

  "&:hover": {

    backgroundColor: red[700],

  },

}));

let LoginSchema = yup.object().shape({

  email: yup

    .string()

    .email("please enter a valid email")

    .required("email is required"),

  password: yup.string().required(),

});

function Copyright(props) {

  return (

    <Typography

      variant="body2"

      color="text.secondary"

      align="center"

      {...props}

    >

      {"Copyright © "}

      <Link color="inherit" href="https://mui.com/">

        Your Website

      </Link>{" "}

      {new Date().getFullYear()}

      {"."}

    </Typography>

  );

}

export default function Login2() {

  const dispatch = useDispatch();

  const dataTa = useSelector((state) => state);

  console.log(dataTa);

  const [showPass, setShowPass] = useState(false);

  const [showbackDrop, setShowbackDrop] = useState(false);

  const handleClickShowPassword = () => {

    setShowPass(!showPass);

  };

  const handleMouseDownPassword = (event) => {

    event.preventDefault();

  };

  const loginUser = async (values) => {

    setShowbackDrop(true);

    const getRes = await axios.post(BASE\_URL + "/api/login", values);

    try {

      const data = getRes;

      console.log(data.data);

      if (data.data.err == "Invalid Credientials") {

        setShowbackDrop(false);

        toast.error(data.data.err);

      } else if (data.data.mag == "user Login SucessFully.") {

        setShowbackDrop(false);

        toast.success("Logged in successfully");

        dispatch({ type: USER\_LOGGED\_IN, payload: data.data });

      } else if (data.data.err == "Invalid Credientials Pass") {

        setShowbackDrop(false);

        toast.error("Invalid Credentials");

      }

    } catch (e) {

      console.log(e);

    }

  };

  const responseGoogle = async (response) => {

    setShowbackDrop(true);

    const res = await response;

    try {

      setShowbackDrop(false);

      console.log(res);

    } catch (e) {

      console.log(e);

    }

  };

  const responseFacebook = (response) => {

    console.log(response);

  };

  return (

    <Grid container component="main" sx={{ height: "100vh" }}>

      <CssBaseline />

      <Grid

        item

        xs={false}

        sm={4}

        md={7}

        sx={{

          backgroundImage: "url(https://source.unsplash.com/random)",

          backgroundRepeat: "no-repeat",

          backgroundColor: (t) =>

            t.palette.mode === "light"

              ? t.palette.grey[50]

              : t.palette.grey[900],

          backgroundSize: "cover",

          backgroundPosition: "center",

        }}

      />

      <Grid item xs={12} sm={8} md={5} component={Paper} elevation={6} square>

        <Box

          sx={{

            my: 8,

            mx: 4,

            display: "flex",

            flexDirection: "column",

            alignItems: "center",

          }}

        >

          <Avatar sx={{ m: 1, bgcolor: "secondary.main" }}>

            <LockOutlinedIcon />

          </Avatar>

          <Typography component="h1" variant="h5">

            Sign in

          </Typography>

          <Container maxWidth={"sm"}>

            <Box

              maxWidth={"sm"}

              mx="auto"

              mt={5}

              //   border={1}

              style={{

                backgroundColor: "white",

                padding: 30,

                boxShadow: 1,

                borderRadius: 7,

              }}

              //   boxShadow={7}

            >

              <Formik

                initialValues={{ email: "", password: "" }}

                validationSchema={LoginSchema}

                validateOnMount={true}

                onSubmit={(values, { setSubmitting }) => {

                  loginUser(values);

                }}

              >

                {({

                  values,

                  errors,

                  touched,

                  handleChange,

                  handleBlur,

                  handleSubmit,

                  isSubmitting,

                  isValid,

                  /\* and other goodies \*/

                }) => (

                  <form onSubmit={handleSubmit}>

                    <Stack spacing={2}>

                      <TextField

                        id="outlined-basic"

                        label="Email"

                        variant="outlined"

                        name="email"

                        fullWidth

                        onChange={handleChange}

                        onBlur={handleBlur}

                        value={values.email}

                      />

                      <Typography

                        variant="p"

                        color={"red"}

                        style={{ margin: 0 }}

                      >

                        {errors.email && touched.email && errors.email}

                      </Typography>

                      <FormControl sx={{ m: 1 }} variant="outlined">

                        <InputLabel htmlFor="outlined-adornment-password">

                          Password

                        </InputLabel>

                        <OutlinedInput

                          id="outlined-adornment-password"

                          type={showPass ? "text" : "password"}

                          value={values.password}

                          onChange={handleChange}

                          onBlur={handleBlur}

                          name="password"

                          fullWidth

                          endAdornment={

                            <InputAdornment position="end">

                              <IconButton

                                aria-label="toggle password visibility"

                                onClick={handleClickShowPassword}

                                onMouseDown={handleMouseDownPassword}

                                edge="end"

                              >

                                {showPass == false ? (

                                  <VisibilityOff />

                                ) : (

                                  <Visibility />

                                )}

                              </IconButton>

                            </InputAdornment>

                          }

                          label="Password"

                        />

                      </FormControl>

                      <Typography

                        variant="p"

                        color={"red"}

                        style={{ margin: 0 }}

                      >

                        {errors.password && touched.password && errors.password}

                      </Typography>

                      <Stack

                        direction={"row"}

                        alignItems={"center"}

                        justifyContent={"space-between"}

                      >

                        <FormGroup>

                          <FormControlLabel

                            control={<Checkbox defaultChecked />}

                            label="Remember me"

                          />

                        </FormGroup>

                        <Typography>Forget password</Typography>

                      </Stack>

                      <Button

                        variant="contained"

                        type="submit"

                        disabled={!isValid}

                      >

                        Login

                      </Button>

                    </Stack>

                  </form>

                )}

              </Formik>

              <Stack>

                <GoogleLogin

                  clientId="1051854074445-ftvvgr3kml6oqv7ujt9902qlbluvrve1.apps.googleusercontent.com"

                  buttonText="Login with google"

                  onSuccess={responseGoogle}

                  render={(renderProps) => (

                    <ColorButton

                      variant="contained"

                      startIcon={<GoogleIcon />}

                      style={{ marginTop: 10 }}

                      onClick={renderProps.onClick}

                      disabled={renderProps.disabled}

                    >

                      Login with Google

                    </ColorButton>

                  )}

                  onFailure={responseGoogle}

                  cookiePolicy={"single\_host\_origin"}

                />

                <FacebookLogin

                  appId="5760285217358093"

                  autoLoad={false}

                  fields="name,email,picture"

                  callback={responseFacebook}

                  icon="fa-facebook"

                  render={(renderProps) => {

                    return (

                      <Button

                        variant="contained"

                        startIcon={<FacebookIcon />}

                        style={{ marginTop: 10 }}

                        onClick={renderProps.onClick}

                      >

                        Login with Facebook

                      </Button>

                    );

                  }}

                />

              </Stack>

              <Grid container mt={5}>

                <Grid item xs>

                  <Link href="#" variant="body2">

                    Forgot password?

                  </Link>

                </Grid>

                <Grid item>

                  <Link

                    to={"/register"}

                    style={{ textDecoration: "none", color: "black" }}

                  >

                    <Typography>

                      Don't have account?

                      <span style={{ color: "blueviolet" }}>Register</span>

                    </Typography>

                  </Link>

                </Grid>

              </Grid>

            </Box>

          </Container>

        </Box>

      </Grid>

    </Grid>

  );

}

import React from "react";

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

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

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

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

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

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

import Visibility from "@mui/icons-material/Visibility";

import VisibilityOff from "@mui/icons-material/VisibilityOff";

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

// import Item from "../../../utils/Items";

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

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

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

import { Container, Typography } from "@mui/material";

import { createTheme, styled } from "@mui/material/styles";

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

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

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

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

import GoogleIcon from "@mui/icons-material/Google";

import FacebookIcon from "@mui/icons-material/Facebook";

import { Formik } from "formik";

import \* as yup from "yup";

import { red } from "@mui/material/colors";

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

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

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

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

import { useState, useEffect } from "react";

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

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

import { ToastContainer, toast } from "react-toastify";

import BASE\_URL from "../../../utils/baseUrl";

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

import ReplayIcon from "@mui/icons-material/Replay";

import axios from "axios";

const label = { inputProps: { "aria-label": "Switch demo" } };

const Item = styled(Paper)(({ theme }) => ({

  ...theme.typography.body2,

  textAlign: "center",

  color: theme.palette.text.secondary,

  height: 60,

  lineHeight: "60px",

}));

const ColorButton = styled(Button)(({ theme }) => ({

  color: theme.palette.getContrastText(red[500]),

  backgroundColor: red[500],

  "&:hover": {

    backgroundColor: red[700],

  },

}));

let RegisterSchema = yup.object().shape({

  email: yup

    .string()

    .email("please enter a valid email")

    .required("email is required"),

  password: yup.string().required(),

});

const Register = () => {

  const navigate = useNavigate();

  const [showPass, setShowPass] = useState(false);

  const [showbackDrop, setShowbackDrop] = useState(false);

  const [showRegis, setShowRegis] = useState(true);

  const [values, setValues] = useState({

    email: "",

    otp: "",

    password: "",

    fname: "",

  });

  const [resendOn, setResendOn] = useState(true);

  const [sentOtp, setSentotp] = useState(false);

  const [verifyOtp, setVerifyotp] = useState(false);

  const [showErr, setShowerr] = useState({

    err: false,

    msg: "",

  });

  const [showErr2, setShowerr2] = useState({

    err: false,

    msg: "",

  });

  const [timeCount, setTimeCount] = useState(5);

  const [timerFun, setTImeFun] = useState(false);

  const handleClickShowPassword = () => {

    setShowPass(!showPass);

  };

  const handleMouseDownPassword = (event) => {

    event.preventDefault();

  };

  const sendOtpWithEmail = async () => {

    setShowbackDrop(true);

    setResendOn(true);

    const sendOtp = await axios.post(BASE\_URL + "/api/emailotp", {

      email: values.email,

    });

    try {

      if (sendOtp.data.msg == "otp sent successfully") {

        setShowbackDrop(false);

        toast.success(sendOtp.data.msg);

        setSentotp(true);

        setTImeFun(true);

        setTimeCount(5);

      } else if (sendOtp.data.msg == "Email Already exist") {

        setShowbackDrop(false);

        toast.warn(sendOtp.data.msg);

        // setSentotp(true);

      }

      console.log(sendOtp);

    } catch (e) {

      console.log(e);

    }

  };

  const verifyOtpWithEmail = async () => {

    setShowbackDrop(true);

    const sendOtp = await axios.post(BASE\_URL + "/api/verifyotp", {

      email: values.email,

      otp: Number(values.otp),

    });

    try {

      if (sendOtp.data.msg == "otp verified successfully") {

        setShowbackDrop(false);

        toast.success(sendOtp.data.msg);

        setVerifyotp(true);

      } else if (sendOtp.data.msg == "invalid otp") {

        setShowbackDrop(false);

        toast.warn(sendOtp.data.msg);

      }

    } catch (e) {

      console.log(e);

    }

  };

  const registerUser = async () => {

    setShowbackDrop(true);

    const sendOtp = await axios.post(BASE\_URL + "/api/register", {

      name: values.fname,

      email: values.email,

      password: values.password,

    });

    try {

      console.log(sendOtp);

      if (sendOtp.data.msg == "User registered") {

        setShowbackDrop(false);

        toast.success("Registered successfully");

        navigate("/login");

      } else {

        setShowbackDrop(false);

        toast.error("Something went wrong");

      }

    } catch (e) {

      console.log(e);

    }

  };

  const checkPassword = (myInput) => {

    // Validate lowercase letters

    var lowerCaseLetters = /[a-z]/g;

    var upperCaseLetters = /[A-Z]/g;

    var numbers = /[0-9]/g;

    var regex = /[`~!@#$%^&\*()\_|+\-=?;:'",.<>\{\}\[\]\\\/]/gi;

    if (!myInput.match(upperCaseLetters)) {

      setShowerr({

        err: true,

        msg: "A uppercase letter (A to Z)",

      });

    } else if (!myInput.match(lowerCaseLetters)) {

      setShowerr({

        err: true,

        msg: "A lowercase letter (a to z)",

      });

    } else if (!myInput.match(numbers)) {

      setShowerr({

        err: true,

        msg: "A number (1 to 100)",

      });

    } else if (!myInput.match(regex)) {

      setShowerr({

        err: true,

        msg: "A special character (!,@,#,$,%,^,&,\*,(,)))",

      });

    } else if (myInput.length < 8) {

      setShowerr({

        err: true,

        msg: "Greater then or equal to 8",

      });

    } else {

      setShowerr({

        err: false,

        msg: "",

      });

    }

    // Validate length

  };

  const checkemail = (val) => {

    const emailReg =

      /^([a-zA-Z0-9\_\-\.]+)@([a-zA-Z0-9\_\-\.]+)\.([a-zA-Z]{2,5})$/;

    if (val.length === 0) {

      setShowerr2({

        err: true,

        msg: "invalid email",

      });

    } else if (!emailReg.test(val)) {

      setShowerr2({

        err: true,

        msg: "invalid email",

      });

    } else {

      setShowerr2({

        err: false,

        msg: "",

      });

    }

  };

  useEffect(() => {

    timerFun == false

      ? null

      : timeCount > 0 && setTimeout(() => setTimeCount(timeCount - 1), 1000);

    return () => clearInterval(timeCount);

  }, [timeCount, timerFun]);

  useEffect(() => {

    timeCount > 0 ? setResendOn(true) : setResendOn(false);

  }, [timeCount, timerFun]);

  console.log(resendOn);

  return (

    <React.Fragment>

      <CssBaseline />

      <Container maxWidth={"sm"}>

        <Box

          maxWidth={"sm"}

          mx="auto"

          mt={5}

          //   border={1}

          style={{

            backgroundColor: "white",

            padding: 30,

            boxShadow: 1,

            borderRadius: 7,

          }}

          boxShadow={7}

        >

          <Box

            mb={4}

            display="flex"

            justifyContent={"center"}

            alignItems="center"

          >

            <Stack direction={"row"} spacing={20}>

              <Typography variant="h5">Register</Typography>

              <FormGroup>

                <FormControlLabel

                  control={

                    <Switch

                      defaultChecked

                      onChange={(e) => setShowRegis(e.target.checked)}

                    />

                  }

                  label="Register with email"

                />

              </FormGroup>

            </Stack>

          </Box>

          {!showRegis ? (

            <Stack>

              <ColorButton

                variant="contained"

                startIcon={<GoogleIcon />}

                style={{ marginTop: 10 }}

              >

                Register with Google

              </ColorButton>

              <Button

                variant="contained"

                startIcon={<FacebookIcon />}

                style={{ marginTop: 10 }}

              >

                Register with Facebook

              </Button>

            </Stack>

          ) : (

            <Stack spacing={2}>

              <TextField

                id="outlined-basic"

                label="Email"

                variant="outlined"

                name="email"

                fullWidth

                placeholder="xyz@gmail.com"

                value={values.email}

                onChange={(e) => {

                  setValues({ ...values, email: e.target.value });

                  checkemail(e.target.value);

                }}

                onBlur={(e) => {}}

              />

              {showErr2.err && (

                <Typography style={{ margin: 0 }} variant="b" color="red">

                  {showErr2.msg}

                </Typography>

              )}

              {sentOtp && (

                <>

                  <TextField

                    id="outlined-basic"

                    label="Otp"

                    variant="outlined"

                    name="otp"

                    fullWidth

                    onChange={(e) =>

                      setValues({ ...values, otp: e.target.value })

                    }

                    value={values.otp}

                    onBlur={(e) => {}}

                  />

                  <Stack direction={"row"} justifyContent={"space-between"}>

                    {timeCount > 0 ? (

                      <Typography>{timeCount}s</Typography>

                    ) : (

                      <Typography>{""}</Typography>

                    )}

                    <Button

                      variant="text"

                      disabled={resendOn}

                      startIcon={<ReplayIcon />}

                      onClick={sendOtpWithEmail}

                    >

                      Resend otp

                    </Button>

                  </Stack>

                </>

              )}

              {verifyOtp && (

                <TextField

                  id="outlined-basic"

                  label="Name"

                  variant="outlined"

                  name="fname"

                  fullWidth

                  onChange={(e) =>

                    setValues({ ...values, fname: e.target.value })

                  }

                  value={values.fname}

                  onBlur={(e) => {}}

                />

              )}

              {verifyOtp && (

                <FormControl sx={{ m: 1 }} variant="outlined">

                  <InputLabel htmlFor="outlined-adornment-password">

                    Password

                  </InputLabel>

                  <OutlinedInput

                    id="outlined-adornment-password"

                    type={showPass ? "text" : "password"}

                    value={values.password}

                    onChange={(e) => {

                      checkPassword(e.target.value);

                      setValues({ ...values, password: e.target.value });

                    }}

                    onBlur={(e) => {}}

                    name="password"

                    fullWidth

                    endAdornment={

                      <InputAdornment position="end">

                        <IconButton

                          aria-label="toggle password visibility"

                          onClick={handleClickShowPassword}

                          onMouseDown={handleMouseDownPassword}

                          edge="end"

                        >

                          {showPass == false ? (

                            <VisibilityOff />

                          ) : (

                            <Visibility />

                          )}

                        </IconButton>

                      </InputAdornment>

                    }

                    label="Password"

                  />

                  {showErr.err && (

                    <Typography variant="b" color="red">

                      {showErr.msg}

                    </Typography>

                  )}

                </FormControl>

              )}

              {verifyOtp == true ? (

                <Button

                  variant="contained"

                  type="submit"

                  size="medium"

                  onClick={registerUser}

                  disabled={showbackDrop}

                  startIcon={

                    showbackDrop && (

                      <CircularProgress size={20} color="inherit" />

                    )

                  }

                >

                  Register

                </Button>

              ) : sentOtp == false ? (

                <Button

                  disabled={

                    values.email.length <= 0

                      ? true

                      : showbackDrop == true

                      ? true

                      : false

                  }

                  variant="contained"

                  type="submit"

                  size="medium"

                  onClick={sendOtpWithEmail}

                  startIcon={

                    showbackDrop && (

                      <CircularProgress size={20} color="inherit" />

                    )

                  }

                >

                  Send Otp

                </Button>

              ) : (

                <Button

                  variant="contained"

                  disabled={showbackDrop}

                  onClick={verifyOtpWithEmail}

                  type="submit"

                  size="medium"

                  startIcon={

                    showbackDrop && (

                      <CircularProgress size={20} color="inherit" />

                    )

                  }

                >

                  Verify otp

                </Button>

              )}

              <Stack

                style={{

                  display: "flex",

                  justifyContent: "center",

                  alignItems: "center",

                  marginTop: 10,

                }}

              >

                <Link

                  to={"/login"}

                  style={{ textDecoration: "none", color: "black" }}

                >

                  <Typography>

                    Already have account?

                    <span style={{ color: "blueviolet" }}>Login</span>

                  </Typography>

                </Link>

              </Stack>

            </Stack>

          )}

        </Box>

      </Container>

      <Backdrop

        sx={{ color: "#fff", zIndex: (theme) => theme.zIndex.drawer + 1 }}

        // open={showbackDrop}

        // onClick={handleClose}

      >

        <CircularProgress color="inherit" />

      </Backdrop>

    </React.Fragment>

  );

};

export default Register;

// import \* as React from "react";

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

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

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

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

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

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

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

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

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

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

import LockOutlinedIcon from "@mui/icons-material/LockOutlined";

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

// function Copyright(props) {

//   return (

//     <Typography

//       variant="body2"

//       color="text.secondary"

//       align="center"

//       {...props}

//     >

//       {"Copyright © "}

//       <Link color="inherit" href="https://mui.com/">

//         Your Website

//       </Link>{" "}

//       {new Date().getFullYear()}

//       {"."}

//     </Typography>

//   );

// }

import React from "react";

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

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

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

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

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

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

import Visibility from "@mui/icons-material/Visibility";

import VisibilityOff from "@mui/icons-material/VisibilityOff";

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

// import Item from "../../../utils/Items";

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

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

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

import { Container, Typography } from "@mui/material";

import { createTheme, styled } from "@mui/material/styles";

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

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

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

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

import GoogleIcon from "@mui/icons-material/Google";

import FacebookIcon from "@mui/icons-material/Facebook";

import { Formik } from "formik";

import \* as yup from "yup";

import { red } from "@mui/material/colors";

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

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

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

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

import { useState, useEffect } from "react";

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

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

import { ToastContainer, toast } from "react-toastify";

import BASE\_URL from "../../../utils/baseUrl";

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

import ReplayIcon from "@mui/icons-material/Replay";

import axios from "axios";

const label = { inputProps: { "aria-label": "Switch demo" } };

const Item = styled(Paper)(({ theme }) => ({

  ...theme.typography.body2,

  textAlign: "center",

  color: theme.palette.text.secondary,

  height: 60,

  lineHeight: "60px",

}));

const ColorButton = styled(Button)(({ theme }) => ({

  color: theme.palette.getContrastText(red[500]),

  backgroundColor: red[500],

  "&:hover": {

    backgroundColor: red[700],

  },

}));

let RegisterSchema = yup.object().shape({

  email: yup

    .string()

    .email("please enter a valid email")

    .required("email is required"),

  password: yup.string().required(),

});

export default function Register2() {

  const navigate = useNavigate();

  const [showPass, setShowPass] = useState(false);

  const [showbackDrop, setShowbackDrop] = useState(false);

  const [showRegis, setShowRegis] = useState(true);

  const [values, setValues] = useState({

    email: "",

    otp: "",

    password: "",

    fname: "",

  });

  const [resendOn, setResendOn] = useState(true);

  const [sentOtp, setSentotp] = useState(false);

  const [verifyOtp, setVerifyotp] = useState(false);

  const [showErr, setShowerr] = useState({

    err: false,

    msg: "",

  });

  const [showErr2, setShowerr2] = useState({

    err: false,

    msg: "",

  });

  const [timeCount, setTimeCount] = useState(5);

  const [timerFun, setTImeFun] = useState(false);

  const handleClickShowPassword = () => {

    setShowPass(!showPass);

  };

  const handleMouseDownPassword = (event) => {

    event.preventDefault();

  };

  const sendOtpWithEmail = async () => {

    setShowbackDrop(true);

    setResendOn(true);

    const sendOtp = await axios.post(BASE\_URL + "/api/emailotp", {

      email: values.email,

    });

    try {

      if (sendOtp.data.msg == "otp sent successfully") {

        setShowbackDrop(false);

        toast.success(sendOtp.data.msg);

        setSentotp(true);

        setTImeFun(true);

        setTimeCount(5);

      } else if (sendOtp.data.msg == "Email Already exist") {

        setShowbackDrop(false);

        toast.warn(sendOtp.data.msg);

        // setSentotp(true);

      }

      console.log(sendOtp);

    } catch (e) {

      console.log(e);

    }

  };

  const verifyOtpWithEmail = async () => {

    setShowbackDrop(true);

    const sendOtp = await axios.post(BASE\_URL + "/api/verifyotp", {

      email: values.email,

      otp: Number(values.otp),

    });

    try {

      if (sendOtp.data.msg == "otp verified successfully") {

        setShowbackDrop(false);

        toast.success(sendOtp.data.msg);

        setVerifyotp(true);

      } else if (sendOtp.data.msg == "invalid otp") {

        setShowbackDrop(false);

        toast.warn(sendOtp.data.msg);

      }

    } catch (e) {

      console.log(e);

    }

  };

  const registerUser = async () => {

    setShowbackDrop(true);

    const sendOtp = await axios.post(BASE\_URL + "/api/register", {

      name: values.fname,

      email: values.email,

      password: values.password,

    });

    try {

      console.log(sendOtp);

      if (sendOtp.data.msg == "User registered") {

        setShowbackDrop(false);

        toast.success("Registered successfully");

        navigate("/login");

      } else {

        setShowbackDrop(false);

        toast.error("Something went wrong");

      }

    } catch (e) {

      console.log(e);

    }

  };

  const checkPassword = (myInput) => {

    // Validate lowercase letters

    var lowerCaseLetters = /[a-z]/g;

    var upperCaseLetters = /[A-Z]/g;

    var numbers = /[0-9]/g;

    var regex = /[`~!@#$%^&\*()\_|+\-=?;:'",.<>\{\}\[\]\\\/]/gi;

    if (!myInput.match(upperCaseLetters)) {

      setShowerr({

        err: true,

        msg: "A uppercase letter (A to Z)",

      });

    } else if (!myInput.match(lowerCaseLetters)) {

      setShowerr({

        err: true,

        msg: "A lowercase letter (a to z)",

      });

    } else if (!myInput.match(numbers)) {

      setShowerr({

        err: true,

        msg: "A number (1 to 100)",

      });

    } else if (!myInput.match(regex)) {

      setShowerr({

        err: true,

        msg: "A special character (!,@,#,$,%,^,&,\*,(,)))",

      });

    } else if (myInput.length < 8) {

      setShowerr({

        err: true,

        msg: "Greater then or equal to 8",

      });

    } else {

      setShowerr({

        err: false,

        msg: "",

      });

    }

    // Validate length

  };

  const checkemail = (val) => {

    const emailReg =

      /^([a-zA-Z0-9\_\-\.]+)@([a-zA-Z0-9\_\-\.]+)\.([a-zA-Z]{2,5})$/;

    if (val.length === 0) {

      setShowerr2({

        err: true,

        msg: "invalid email",

      });

    } else if (!emailReg.test(val)) {

      setShowerr2({

        err: true,

        msg: "invalid email",

      });

    } else {

      setShowerr2({

        err: false,

        msg: "",

      });

    }

  };

  useEffect(() => {

    timerFun == false

      ? null

      : timeCount > 0 && setTimeout(() => setTimeCount(timeCount - 1), 1000);

    return () => clearInterval(timeCount);

  }, [timeCount, timerFun]);

  useEffect(() => {

    timeCount > 0 ? setResendOn(true) : setResendOn(false);

  }, [timeCount, timerFun]);

  return (

    <Grid container component="main" sx={{ height: "100vh" }}>

      <CssBaseline />

      <Grid

        item

        xs={false}

        sm={4}

        md={7}

        sx={{

          backgroundImage: "url(https://source.unsplash.com/random)",

          backgroundRepeat: "no-repeat",

          backgroundColor: (t) =>

            t.palette.mode === "light"

              ? t.palette.grey[50]

              : t.palette.grey[900],

          backgroundSize: "cover",

          backgroundPosition: "center",

        }}

      />

      <Grid item xs={12} sm={8} md={5} component={Paper} elevation={6} square>

        <Box

          sx={{

            my: 8,

            mx: 4,

            display: "flex",

            flexDirection: "column",

            alignItems: "center",

          }}

        >

          <Avatar sx={{ m: 1, bgcolor: "secondary.main" }}>

            <LockOutlinedIcon />

          </Avatar>

          <Typography component="h1" variant="h5">

            Sign up

          </Typography>

          <Container maxWidth={"sm"}>

            <Box

              maxWidth={"sm"}

              mx="auto"

              mt={5}

              //   border={1}

              style={{

                backgroundColor: "white",

                padding: 30,

                boxShadow: 1,

                borderRadius: 7,

              }}

              //   boxShadow={7}

            >

              <FormGroup>

                <FormControlLabel

                  control={

                    <Switch

                      defaultChecked

                      onChange={(e) => setShowRegis(e.target.checked)}

                    />

                  }

                  label="Register with email"

                />

              </FormGroup>

              {!showRegis ? (

                <Stack>

                  <ColorButton

                    variant="contained"

                    startIcon={<GoogleIcon />}

                    style={{ marginTop: 10 }}

                  >

                    Register with Google

                  </ColorButton>

                  <Button

                    variant="contained"

                    startIcon={<FacebookIcon />}

                    style={{ marginTop: 10 }}

                  >

                    Register with Facebook

                  </Button>

                </Stack>

              ) : (

                <Stack spacing={2}>

                  <TextField

                    id="outlined-basic"

                    label="Email"

                    variant="outlined"

                    name="email"

                    fullWidth

                    placeholder="xyz@gmail.com"

                    value={values.email}

                    onChange={(e) => {

                      setValues({ ...values, email: e.target.value });

                      checkemail(e.target.value);

                    }}

                    onBlur={(e) => {}}

                  />

                  {showErr2.err && (

                    <Typography style={{ margin: 0 }} variant="b" color="red">

                      {showErr2.msg}

                    </Typography>

                  )}

                  {sentOtp && (

                    <>

                      <TextField

                        id="outlined-basic"

                        label="Otp"

                        variant="outlined"

                        name="otp"

                        fullWidth

                        onChange={(e) =>

                          setValues({ ...values, otp: e.target.value })

                        }

                        value={values.otp}

                        onBlur={(e) => {}}

                      />

                      <Stack direction={"row"} justifyContent={"space-between"}>

                        {timeCount > 0 ? (

                          <Typography>{timeCount}s</Typography>

                        ) : (

                          <Typography>{""}</Typography>

                        )}

                        <Button

                          variant="text"

                          disabled={resendOn}

                          startIcon={<ReplayIcon />}

                          onClick={sendOtpWithEmail}

                        >

                          Resend otp

                        </Button>

                      </Stack>

                    </>

                  )}

                  {verifyOtp && (

                    <TextField

                      id="outlined-basic"

                      label="Name"

                      variant="outlined"

                      name="fname"

                      fullWidth

                      onChange={(e) =>

                        setValues({ ...values, fname: e.target.value })

                      }

                      value={values.fname}

                      onBlur={(e) => {}}

                    />

                  )}

                  {verifyOtp && (

                    <FormControl sx={{ m: 1 }} variant="outlined">

                      <InputLabel htmlFor="outlined-adornment-password">

                        Password

                      </InputLabel>

                      <OutlinedInput

                        id="outlined-adornment-password"

                        type={showPass ? "text" : "password"}

                        value={values.password}

                        onChange={(e) => {

                          checkPassword(e.target.value);

                          setValues({ ...values, password: e.target.value });

                        }}

                        onBlur={(e) => {}}

                        name="password"

                        fullWidth

                        endAdornment={

                          <InputAdornment position="end">

                            <IconButton

                              aria-label="toggle password visibility"

                              onClick={handleClickShowPassword}

                              onMouseDown={handleMouseDownPassword}

                              edge="end"

                            >

                              {showPass == false ? (

                                <VisibilityOff />

                              ) : (

                                <Visibility />

                              )}

                            </IconButton>

                          </InputAdornment>

                        }

                        label="Password"

                      />

                      {showErr.err && (

                        <Typography variant="b" color="red">

                          {showErr.msg}

                        </Typography>

                      )}

                    </FormControl>

                  )}

                  {verifyOtp == true ? (

                    <Button

                      variant="contained"

                      type="submit"

                      size="medium"

                      onClick={registerUser}

                      disabled={showbackDrop}

                      startIcon={

                        showbackDrop && (

                          <CircularProgress size={20} color="inherit" />

                        )

                      }

                    >

                      Register

                    </Button>

                  ) : sentOtp == false ? (

                    <Button

                      disabled={

                        values.email.length <= 0

                          ? true

                          : showbackDrop == true

                          ? true

                          : false

                      }

                      variant="contained"

                      type="submit"

                      size="medium"

                      onClick={sendOtpWithEmail}

                      startIcon={

                        showbackDrop && (

                          <CircularProgress size={20} color="inherit" />

                        )

                      }

                    >

                      Send Otp

                    </Button>

                  ) : (

                    <Button

                      variant="contained"

                      disabled={showbackDrop}

                      onClick={verifyOtpWithEmail}

                      type="submit"

                      size="medium"

                      startIcon={

                        showbackDrop && (

                          <CircularProgress size={20} color="inherit" />

                        )

                      }

                    >

                      Verify otp

                    </Button>

                  )}

                  <Stack

                    style={{

                      display: "flex",

                      justifyContent: "center",

                      alignItems: "center",

                      marginTop: 10,

                    }}

                  >

                    <Link

                      to={"/"}

                      style={{ textDecoration: "none", color: "black" }}

                    >

                      <Typography>

                        {"Already have account?"}

                        <span style={{ color: "blueviolet" }}>Login</span>

                      </Typography>

                    </Link>

                  </Stack>

                </Stack>

              )}

            </Box>

          </Container>

          {/\* <Box

            component="form"

            noValidate

            onSubmit={handleSubmit}

            sx={{ mt: 1 }}

          >

            <TextField

              margin="normal"

              required

              fullWidth

              id="email"

              label="Email Address"

              name="email"

              autoComplete="email"

              autoFocus

            />

            <TextField

              margin="normal"

              required

              fullWidth

              name="password"

              label="Password"

              type="password"

              id="password"

              autoComplete="current-password"

            />

            <FormControlLabel

              control={<Checkbox value="remember" color="primary" />}

              label="Remember me"

            />

            <Button

              type="submit"

              fullWidth

              variant="contained"

              sx={{ mt: 3, mb: 2 }}

            >

              Sign In

            </Button>

            <Grid container>

              <Grid item xs>

                <Link href="#" variant="body2">

                  Forgot password?

                </Link>

              </Grid>

              <Grid item>

                <Link href="#" variant="body2">

                  {"Don't have an account? Sign Up"}

                </Link>

              </Grid>

            </Grid>

            <Copyright sx={{ mt: 5 }} />

          </Box> \*/}

        </Box>

      </Grid>

    </Grid>

  );

}

### Implementation

Chapter 5

### Project implementation plan includes :

* + Once the user opens a website, he is asked to enable location. The location of every place is stored with latitude and longitude in the database**.**

### Other steps are:

* + How user gets information about the institution?
  + How details about the school are uploaded?
  + Admin dashboard

### How user gets information about the institution?

* User is asked to select the range of distance up to which he is looking for the institution. The area within the circumference of that range as a radius will be covered.
* Now the home page will show a banner of the preferred type of institution with top ratings. Priorities can be set in the filter menu. Eg - if a user is only interested to see schools, then he can use the filter option to school option.
* if a person is looking for coaching for a particular subject, he can select the type of institute as coaching and preferred subject as commerce in the filter section. The home page also has a map, which shows institutes on it. Users can navigate it.

### How details about schools are uploaded?

* User who wants to upload details about the institute doesn’t necessarily have to be from the management/ owner of the institution. Any local person can do this by the register and sign-up options.
* Once all the available information is provided, it gets stored in the database and remains there until it is verified.
* The team visits institutions or contacts remotely to get the required formal documents for verification purposes. If an institution passes the verification, then it is uploaded on the website, otherwise rejected.
* the institution also signs up in the owner category. They get their separate dashboard to control the content and information they are uploading. It is done through the content management system.
* The management of schools can also customize the way the page about the institution appears. Nobody can interfere with data provided by another party. Suppose a student provides feedback. Although institution management has a separate dashboard they have no access to remove feedback provided by students.

## ADMIN DASHBOARD

* The owner of the website has an admin dashboard that controls everything.
* all the modules are managed through secured APIs.
* Eg- When verification of an institution is done, all the data which has been stored in an organized way in the database gets uploaded just by a single click on the radio button.
* Admin can only verify the school. He doesn’t have to manage data provided by a different set of users.

A separate content management system built for admin manages everything automatically and dynamically

## ADMIN DASHBOARD

* The owner of the website has an admin dashboard that controls everything.
* all the modules are managed through secured APIs.
* Eg- When verification of an institution is done, all the data which has been stored in an organized way in the database gets uploaded just by a single click on the radio button.
* Admin can only verify the school. He doesn’t have to manage data provided by a different set of users.
* A separate content management system built for the admin manages everything automatically and dynamically**.**

**CONCLUSION**

Using the website, user of type one category that is student searches provides access to the location. Then on the basis of location, he gets listing of all of the institute nearby his area. This data get retrieved from the MongoDB database with help of the APIs. User can also choose to write pincode of the place he wants to search a institute for. Now there are various sets of filers he can apply on the showcasing of card about institute like filter for fees can be applied, filter for type of boards can be applied, filter for specification in a subject can be applied, filter for availability of particular facility including sports, hostel, NCC exposure etc can be applied. Now result will vary according to the type of filer applied. The card of institute shows name of institute, it picture, and rating. When user navigates a card, he reaches a separate page showing all of the details about institute and all content provided by it. That specific page is managed by the institute only. Those students who are already admitted to the institute can provide review. Those review help another finders to get

### Security

* **Data is exchanged between the frontend and backend through highly secure API’s so that no external party can perform unverified changes. After registration, while signing up, the user doesn’t type in a password.**
* **He provides his email id and then he receives an autogenerated password on email, which he has to use while logging in. The website has highly secure access management.**
* **School management is given a different categorial verification. No party has access to the data provided by other parties. Two-factor verification for school management.**

Images

