**Mobile Payment Application & Website**

**A PROJECT REPORT Submitted**

**in partial fulfillment of the Requirements**

**for the Degree of**

**MASTER of COMPUTER APPLICATIONS**

by

**Akshay Kumar Maurya**

**University Roll. No.**

**1900290149010**

**Submitted to**

**Dr. Sangeeta Arora**

**(AssociateProfessor)**

**KIET Group of Institutions, Ghaziabad**



to the

**Department of Computer Applications,**

**Dr A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY**

**LUCKNOW**

**(Formerly Uttar Pradesh Technical University, Lucknow)**

**(July 2021)**

**DECLARATION**

I hereby declare that the work presented in this report entitled “Mobile Payment Application & Website”, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Akshay Kumar Maurya

Roll. No.: 1900290149010

Branch: Master of Computer Applications

**(Candidate Signature)**



**CERTIFICATE**

Certified that **Akshay Kumar Maurya** (**1900290149010**) has carried out the project work presented in this report entitled “**Mobile Payment Application & Website**” for the award of **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

**Dr. Sangeeta Arora External Examiner**

Associate Professor

Dept. of Computer Applications

KIET Group of Institutions, Ghaziabad

**Dr. Ajay Kumar Srivastava**

Professor & Head

Department of Computer Application

KIET Group of Institutions, Ghaziabad

Date:

**ACKNOWLEDGEMENTS**

Firstly, I thank **MOBILOITTE** for allowing me to complete my project

successfully. I express my sincere gratitude to all those who initiated and helped me in the successful completion of this project. Sincere thanks and profound gratitude to my project Manager **Mr. Prabhash Singh** for their strict supervision, constant encouragement, inspiration and guidance, which ensure the worthiness of my work. Working under them was an enrich experience and my project Internal Supervisor **Dr. Sangeeta Arora** for helping in carrying out the project work and for many valuable and useful information and guidance while bringing out this project.

I am highly grateful to **Dr. Ajay Kumar Shrivastava**, **HOD** of **MCA** KIET Group Of

Institutions, Ghaziabad for providing this opportunity to carry out the Major Project at

**“Mobile Payment Application & Website”**. KIET Group Of Institutions Ghaziabad for providing academic inputs, guidance & Encouragement throughout this period.

Finally, I express my indebtedness to all who have directly or indirectly contributed to the successful completion of my major project.

Akshay Kumar Maurya

1900290149010

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Declaration ii  Certificate iii  Acknowledgements iv  Table of Contents v-vi | |  |
|  |  | |  |
|  | **CHAPTER 1: INTRODUCTION 7-10** | |
|  | 1.1 Project Description 7 | |
|  | 1.2 Project Scope 8 | |
|  | 1.3 Hardware/Software used in Project 9  1.3.1 Technology Description 10 | |
|  | **CHAPTER 2: LITERATURE REVIEW 11-16** | |  |
|  |  | 2.1 Create Custom Object 11  2.1.1 Sing Up 12  2.1.2 Login 13 |  |
|  |  | 2.2 Operational feasibility 14 |  |
|  |  | 2.3 Economic feasibility 15 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 2.4 Behavioural feasibility 16 | |  | |
| **CHAPTER 3: DATABASE MANAGEMENT 17** | | |  | |
| 3.1 Server-side  3.2 Client-side | | | 17  17 | |
| **CHAPTER 4: OBJECT INTERFACES/Coding** | | | **18-90** | |
| 4.1 Interface (Screenshot)/Coding | | | 18-90 | |
| CHAPTER 5: Extend Your Mobile and Reactive Apps Using JavaScript | | | **91-92** | |
| 5.1 Website Setup | | | 91-92 | |
| **CHAPTER 6: TESTING 93-97**  6.1 Exploratory Testing 93  6.2 Test Automation 94  6.3 Load Testing 95 6.4Security Testing 96-97 | | |  | |

**CHAPTER 7: REFERENCES 98-99**

**CHAPTER 1**

### INTRODUCTION

In this chapter we will give an introduction about security locks and protection on computer system. Here we also give an overview of how the data and file are encrypted and decrypted in system.

### 1.1 Project Description

The project is based on “**cryptocurrency”.** A cryptocurrency is a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology—a distributed ledger enforced by a disparate network of computers.

Some of cryptocurrency are available in market **:-**

* Bitcoin
* Litecoin
* Ethereum
* Ripple
* NEO etc.

### 1.2 Project Scope

The scope of cryptocurrency is bright as a lot of developments have led to the growth of the cryptocurrency industry. Some of the highlights are:

* Big banks, companies, and institutional investors such as JP Morgan, MicroStrategy, Tesla, and PayPal have entered the space
* There are over 9,000 cryptocurrencies in existence today
* The cryptocurrency market capcrossed the $2 trillion mark
* **Emerging cryptocurrency regulations by governments across the world, formalizing the industry, lowering risks associated with crypto investments**
* The emergence of Central Bank Digital Currency (CBDC)
* **Bitcoin being compared to gold as a store of value**
* Innovative developments such as [Decentralised Finance](https://dcxlearn.com/trading/what-is-decentralized-finance-defi-2/) (#DeFi) and Non-Fungible Tokens (NFTS)

There is increasing adoption as governments across the world have started regulating the cryptocurrency industry and India too is embracing regulations in the crypto space. As the largest cryptocurrency exchange in India, we at CoinDCX aim to be at the forefront of this revolution. Through our simple, user-friendly app, [CoinDCX GO](http://app.adjust.net.in/17agule?campaign=CampaignName&adgroup=AdgroupName&creative=CreativeName)[,](https://apps.apple.com/app/id1517787269?mt=8) you can easily enter the market and start investing.

* 1. **Hardware / Software used in the Project**

**Table 1.1 Hardware**

|  |  |
| --- | --- |
| **Hardware** | **Configuration** |
| Processor | Intel Pentium G2030 clocked at 3.0 GHz |
| RAM | 4GB DDR3 |
| Monitor | Dell Backlit 21” LED |
| Modem | Internet Connectivity |
| Keyboard | Dell Standard 102 Keys & Optical Mouse |

**Table 1.2 Software**

|  |  |
| --- | --- |
| **Software** | **Configuration** |
| Operating System | Windows XP /7/8/10 & mac os |

|  |  |
| --- | --- |
| Software | Chrome, Microsoft Edge, Safari |

### 1.3.1 Technology Description

**JavaScript**

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.).  There are also more advanced server side versions of JavaScript such as Node.js, which allow you to add more functionality to a website than downloading files (such as realtime collaboration between multiple computers). Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example:

* *Client-side JavaScript* extends the core language by supplying objects to control a browser and its *Document Object Model* (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.
* *Server-side JavaScript* extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

This means that in the browser, JavaScript can change the way the webpage (DOM) looks. And, likewise, Node.js JavaScript on the server can respond to custom requests from code written in the browser.

**CHAPTER 2**

**LITERATURE REVIEW**

Considerations for the project:

Some of the requirements are:

1. To have list of diﬀerent types of cryptocurrence available in the market.
2. Track all the pending and complete requests of user and get the status of the booking.
3. Take the requests from user and admin members and prioritize in buying product.
4. Give proper security accesses to product so that pending requests are not edited and deleted by everyone.
5. Inform admin member when the product requested arrives.
6. Include Reports to generate defaulters list, cryptocurrency collected every booking.
   1. **Create the following Custom Objects**

**2.1.1 SingUp** – This object holds all the information related to a user.

**Table no.2.1.1 SingUp**

|  |  |  |  |
| --- | --- | --- | --- |
| Data Type | Field Label | Other Values | Remarks |
| Text | user\_name |  |  |
| Text | User\_email |  | Email validation are require. |
| Picklist | User\_type | Individual  Corporate | Mandatory |
| Text | Password |  | Password validation require. |
| Number | Phone number |  | Phone number (Mandatory) |

* + 1. **Login** – This object holds all the information related to a user.

**Table no.2.1.2 Login**

|  |  |  |  |
| --- | --- | --- | --- |
| Data Type | Field Label | Other Values | Remarks |
| Text | Email | Length:50 | Member’s Email – Mandatory |
| Text | password | Length:50 | Member’s password – Mandatory |

1.On SingUp, screen – on the singup screen need to fill Member data who want to login in the website.

2. On clicking “Register Now” it send an OTP to register email and after enter the OTP user easly go to login screen and login the screen.

3. On Login screen enter the email and password which are given when user singup time.

4. Both are required ﬁelds.

5. Once the login is successful, need to show the all the data of the website on the dashboard screen.

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities.

**2.2. Technical Feasibility**

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

Does the existing technology sufficient for the suggested one?

Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within latest technology. Through the technology may become obsolete after some period of time, due to the fact that never version of same software supports older versions, the system may still be used. So, there are minimal constraints involved with this project. The system has been developed using Java the project is technically feasible for development.

**2.3. Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:

 Is there sufficient support for the management from the users?

Will the system be used and work properly if it is being developed and implemented?

Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits.

**2.4. Economic Feasibility**

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

The costs conduct a full system investigation.

The cost of the hardware and software.

The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, it gives an indication of the system is economically possible for development.

**2.4. Behavioral Feasibility**

An estimate should be made of how strong a reaction the user staff is likely to have towards the development of a computerized system. It is common knowledge that computer installation have something to do with Turnover, Transfers and changes in employee Job Status. Normal human psychology of human beings indicate that people are resistant to change and computers are known to facilitate change. Any project formulations should consider this factor also. Before the development of the Project titled "Delhi Metro", the need to study the feasibility of the successful execution of the project was felt and thus the following factors are considered for a Feasibility Study.

1. Need Analysis.
2. Provide the users information pertaining to the preceding requirement.

**CHAPTER 3**

**DATABASE MANAGEMENT**

**3.1. Server-Side**

The APIs for developing applications using the DHTML client in DB2 Alphablox are available on the server-side, where a developer accesses them through Java™ calls (for example, in a Java scriptlet on a JSP page). The reason the Java APIs are called server-side APIs is because the code executes on the server before it is sent to the browser.

Executing code on the server is often more efficient, and also makes it easier to create web pages that work correctly on multiple browsers. The DHTML client is designed to keep the client and the server in sync without page refreshing. When you execute code on the server, only affected areas in the Blox UI is refreshed, not the whole page.

### 3.2. Client-Side

There are also times when you want to use the DHTML client’s Client API for tasks that are best handled on the client. These are called client-side APIs because they are interpreted by the browsers. Often times you want to call some server-side code to change Blox properties on the server via some JavaScript™ code on the client when a user clicks a button or link on the page.

The DHTML client has a relatively straightforward API on the client-side.

**CHAPTER 4**

**OBJECT INTERFACES/Coding**

### 

### 4.1 SingUp Screen:-

This is singup screen where user can register them for login.

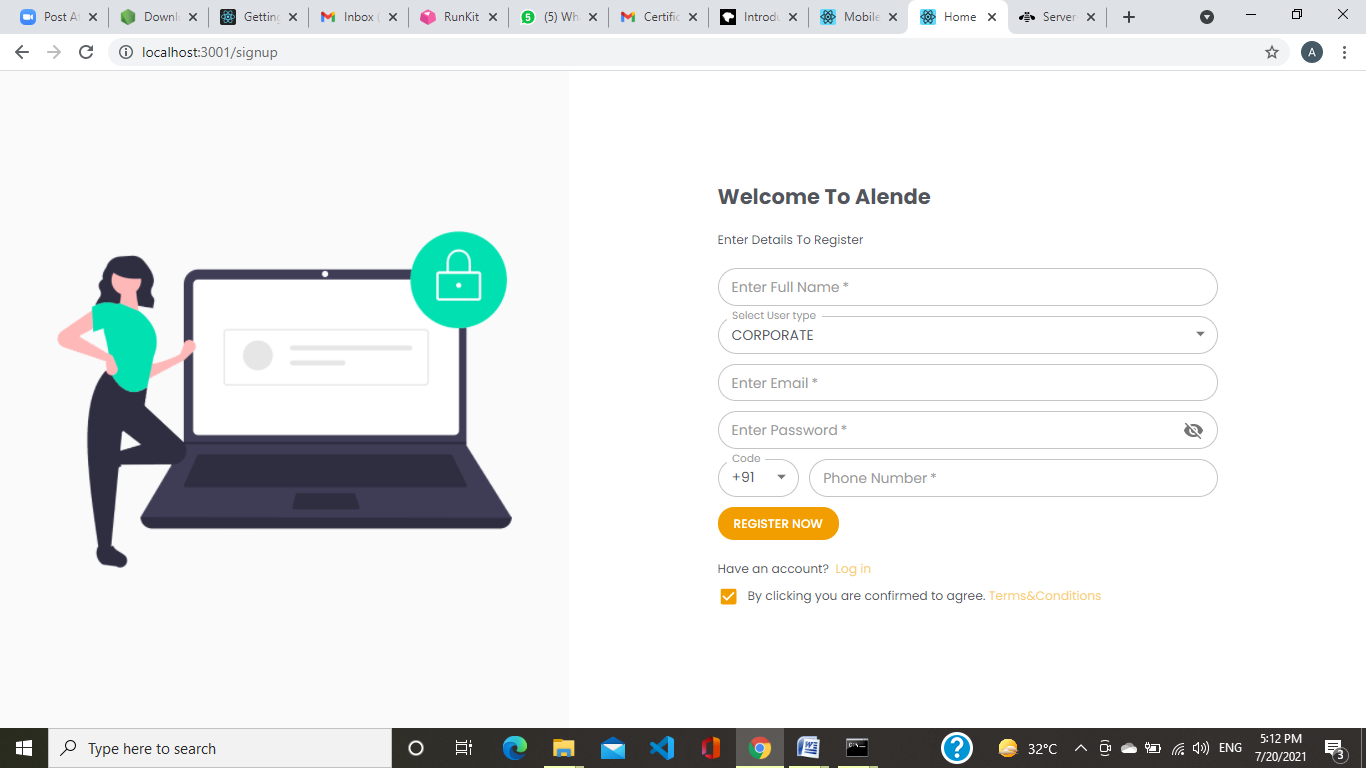
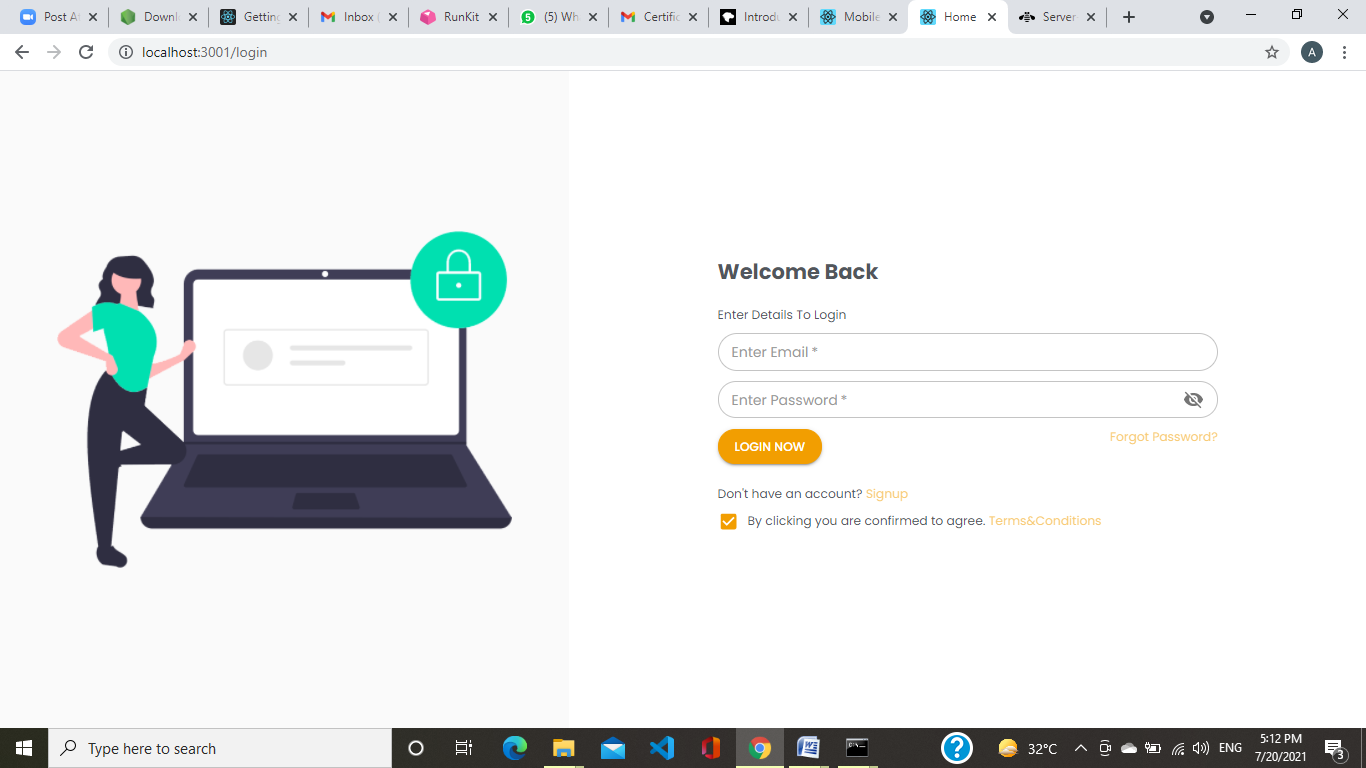


Fig. 4.1

### 4.2 Login Screen:-

This is login screen where user can login for this website.



### Fig. 4.2

### 4.3 Dashboard Screen:-

This is dashboard screen where user can see their cryptocurrency balance and purchase product.

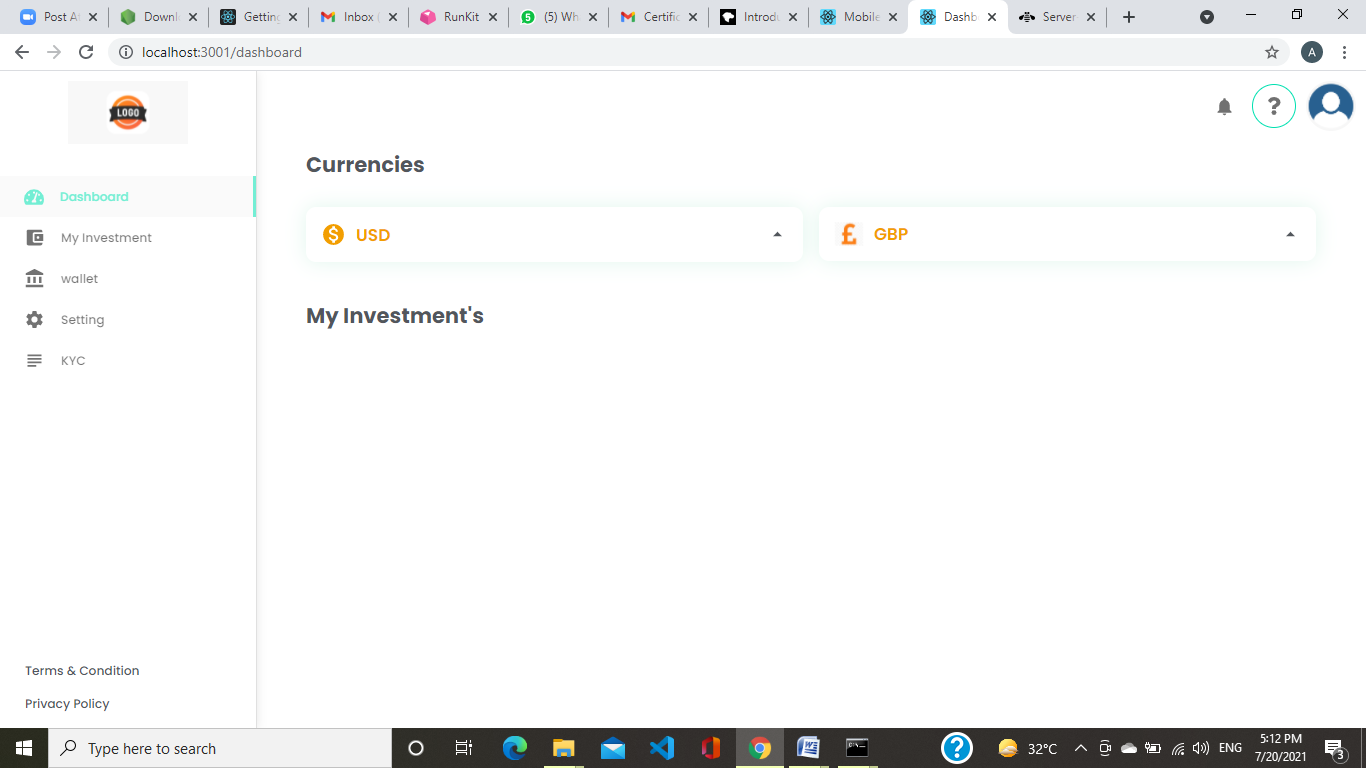


Fig. 4.3

.

### 4.4 My Investment Screen:-

This is investment screen where user can see their all purchase

Product.

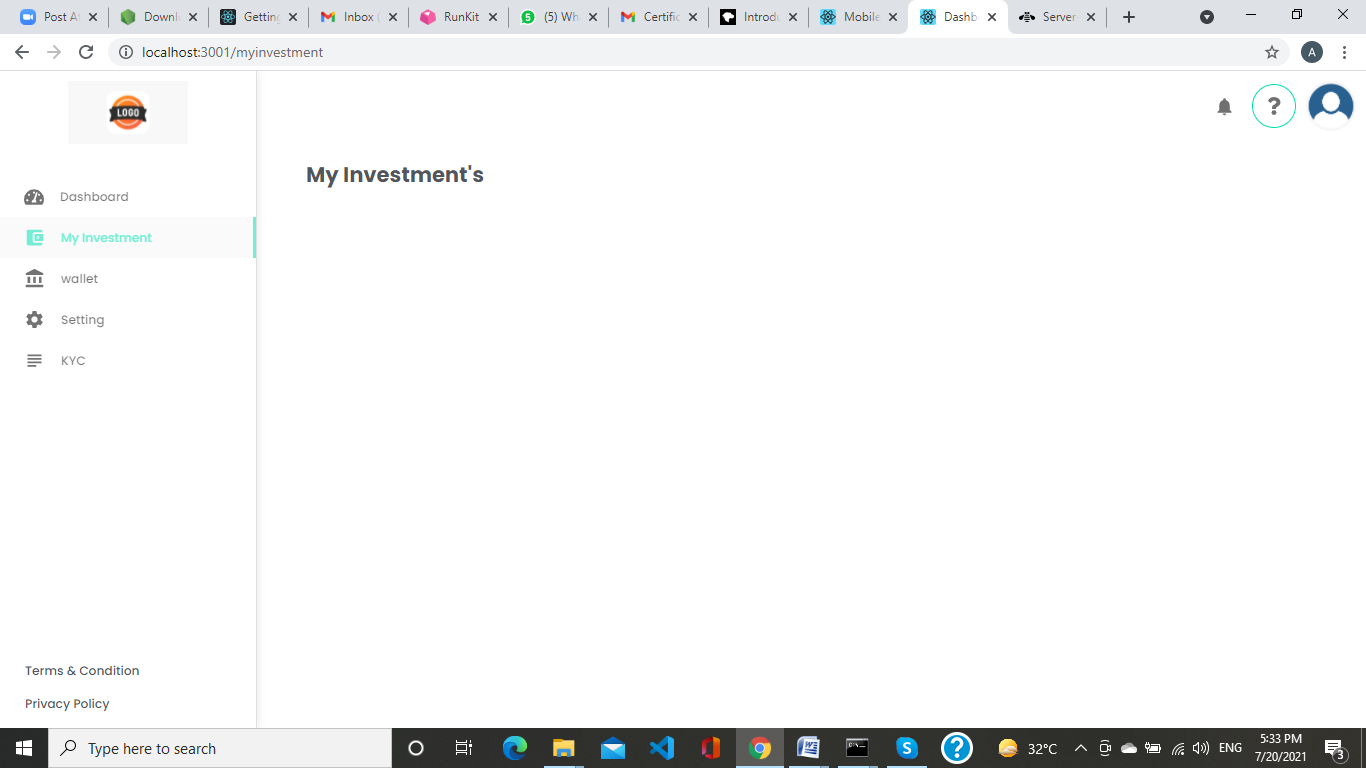


Fig. 4.4

4.5 Wallet Screen:-

This is wallet screen where user can see their wallet balance and latest transaction of the user.

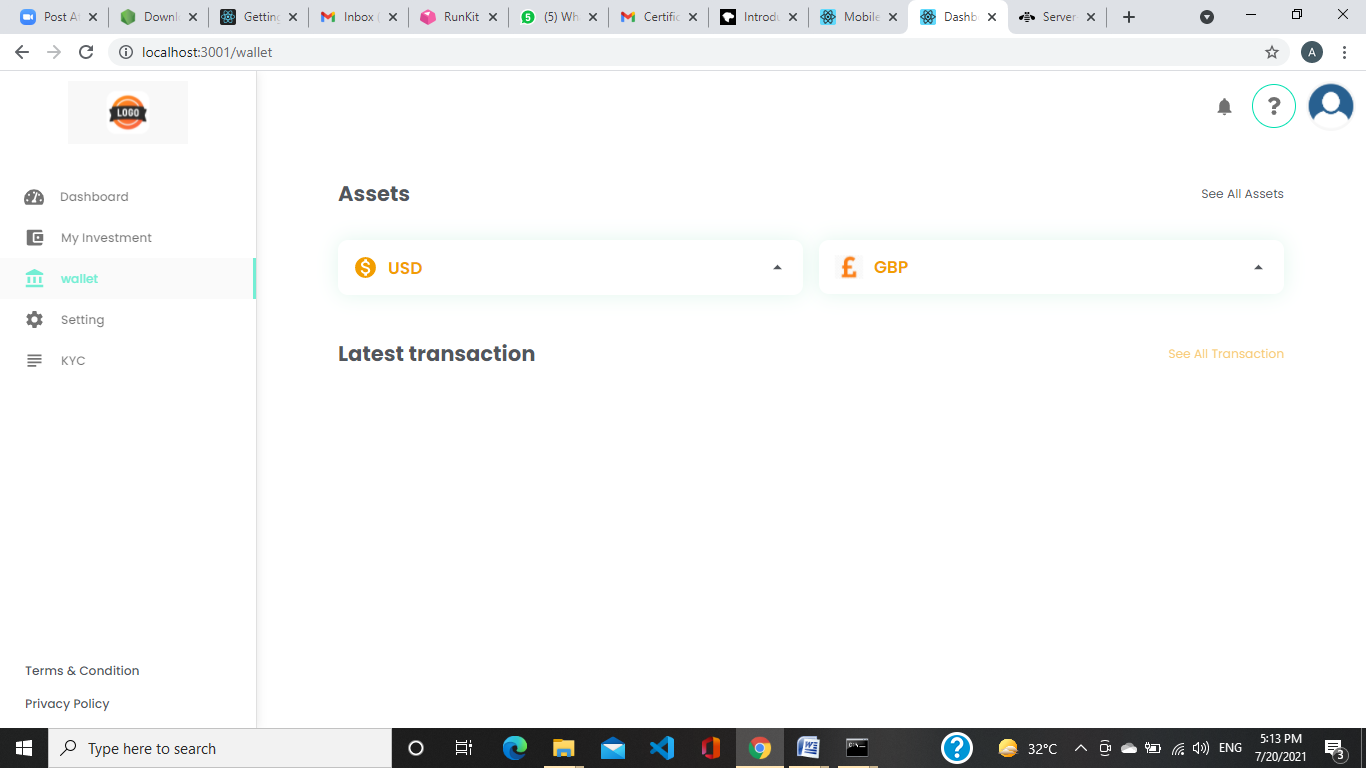


Fig. 4.5

### 4.6. See All Transection Screen:-

This is the all transaction screen where user can see

the their all transaction done.

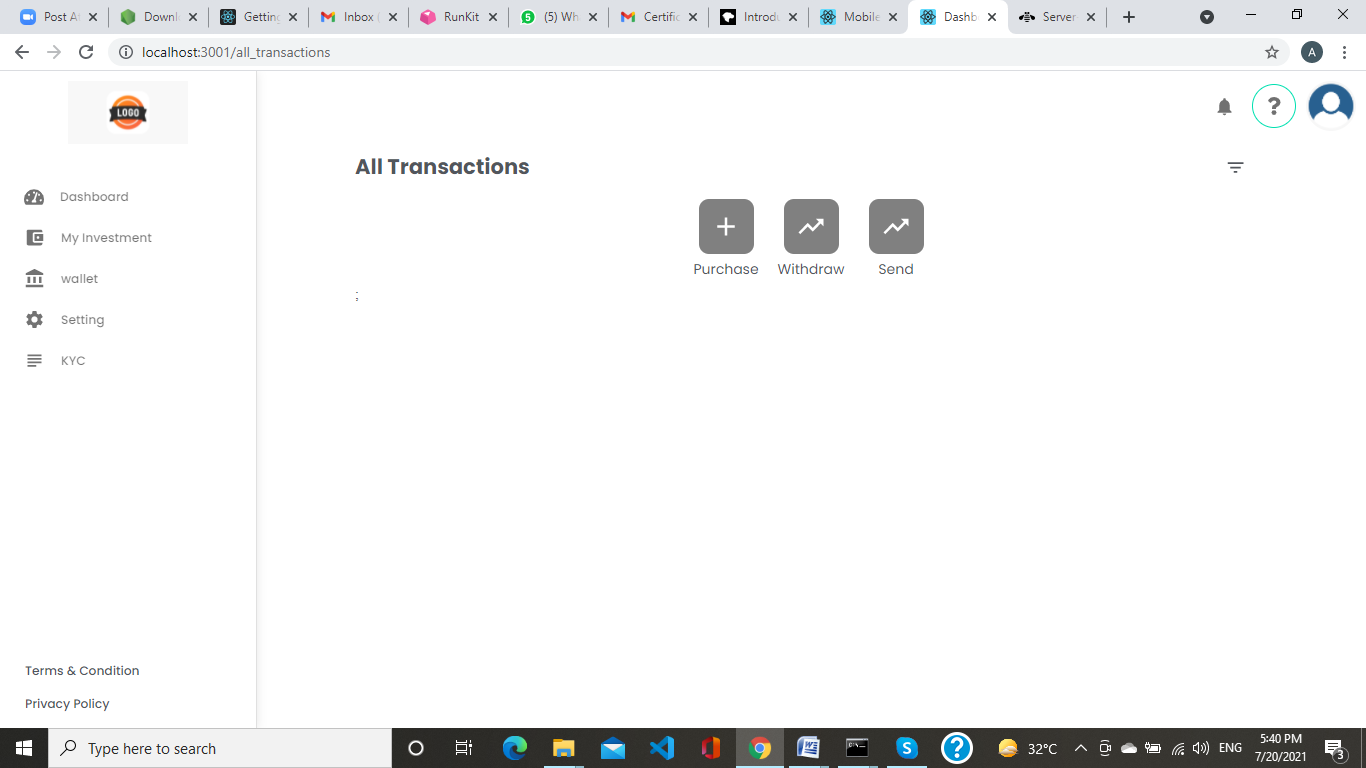


Fig. 4.6

4.7. Purchase Screen:-

This is purchase screen where user can purchase their product.

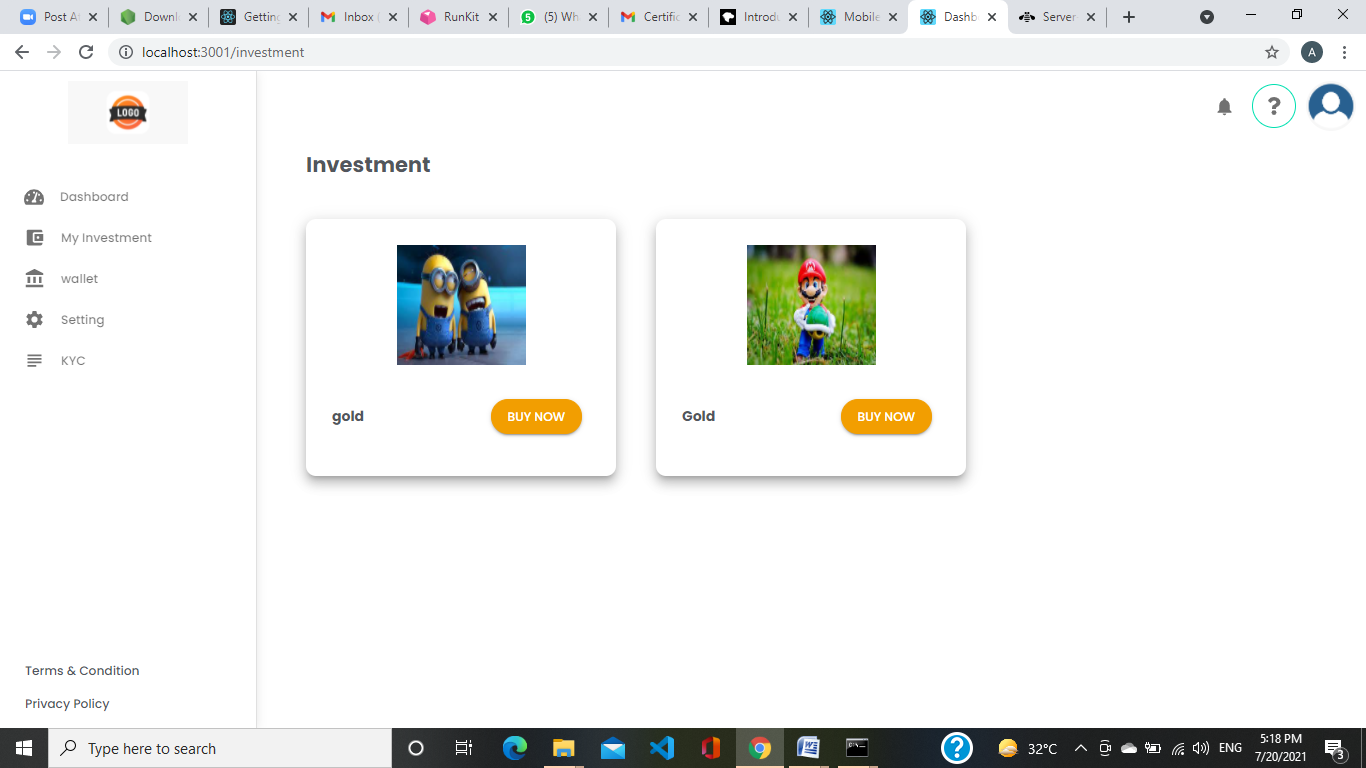


Fig. 4.7

### 4.8. Buy Screen:-

This is buy screen where user can buy the product by using their currency.

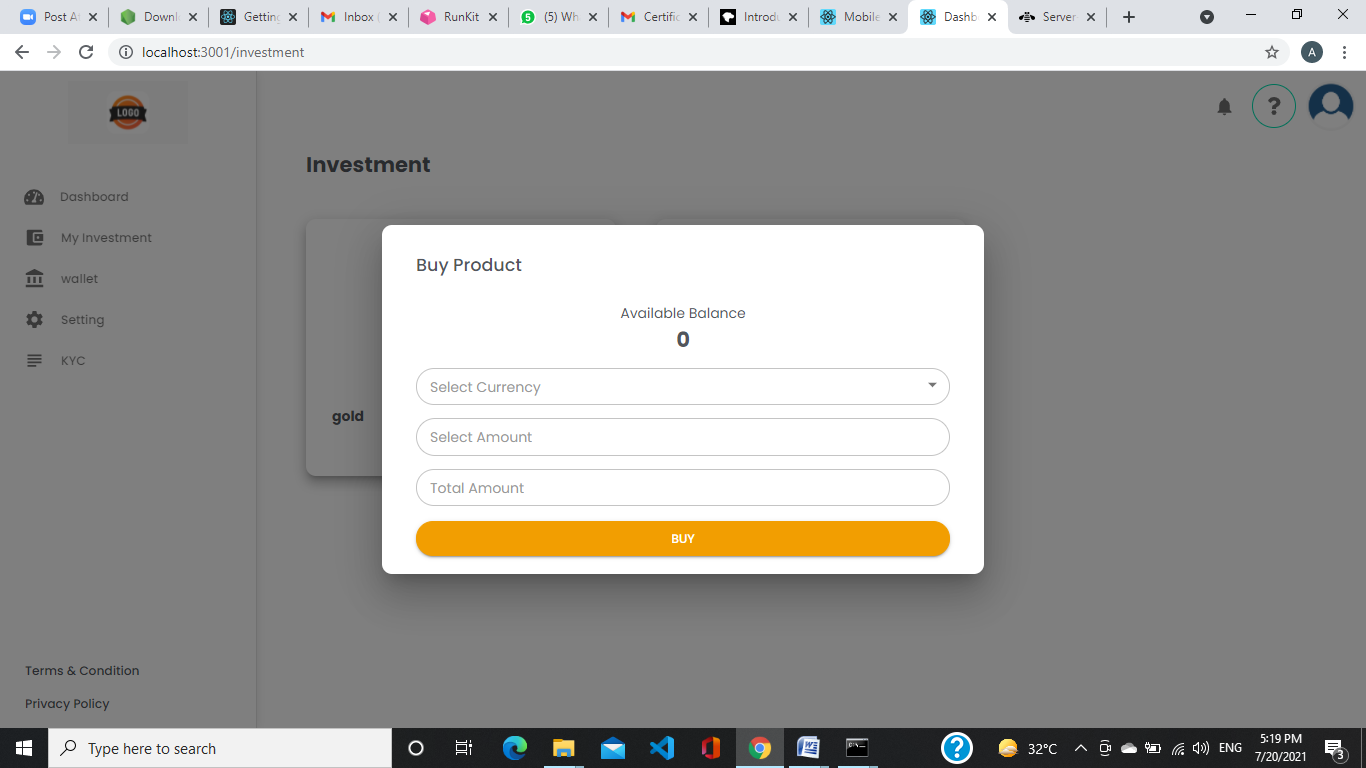


Fig. 4.8

### 4.9. Withdrawn Screen:-

This is withdrawn screen where user can withdraw thir currency.

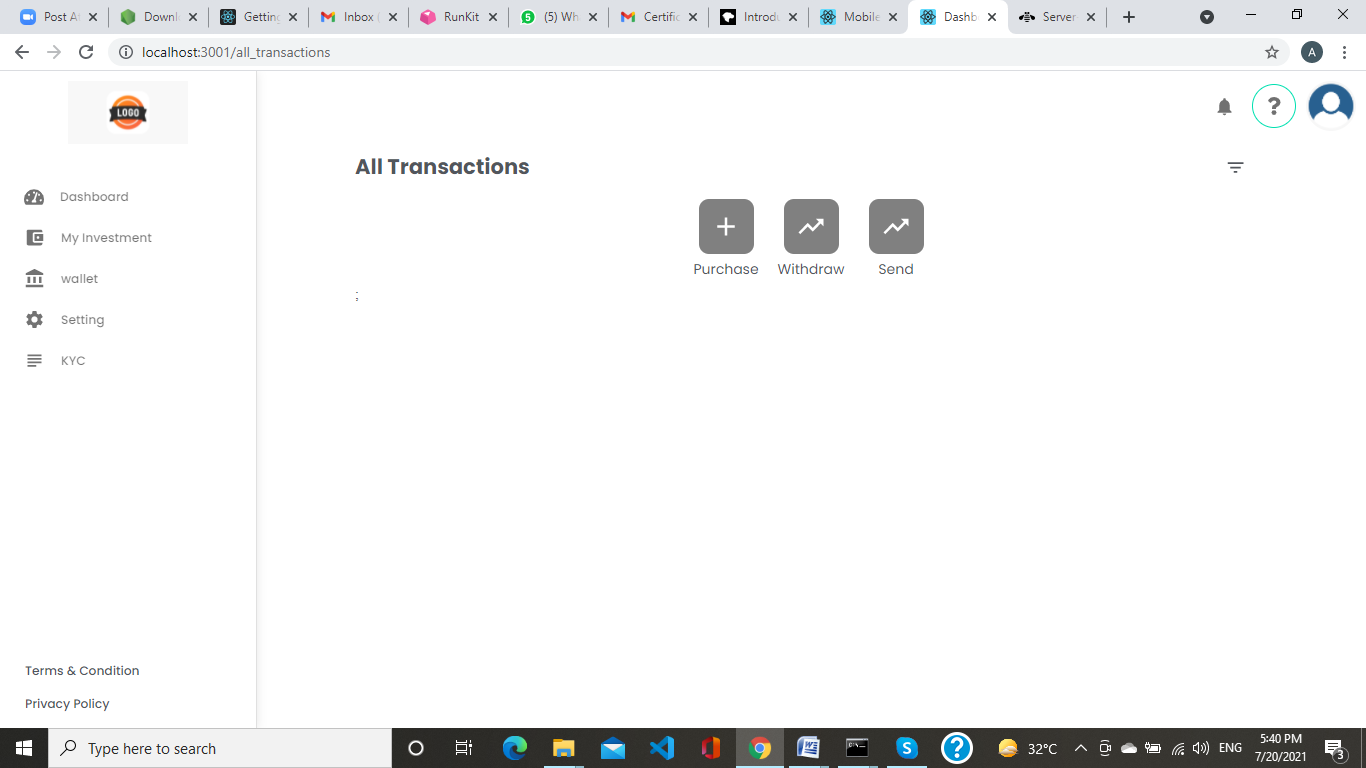


Fig. 4.9

### 4.10. Withdrawn Screen:-

This is withdrawn screen where user can select which currency want to withdraw.

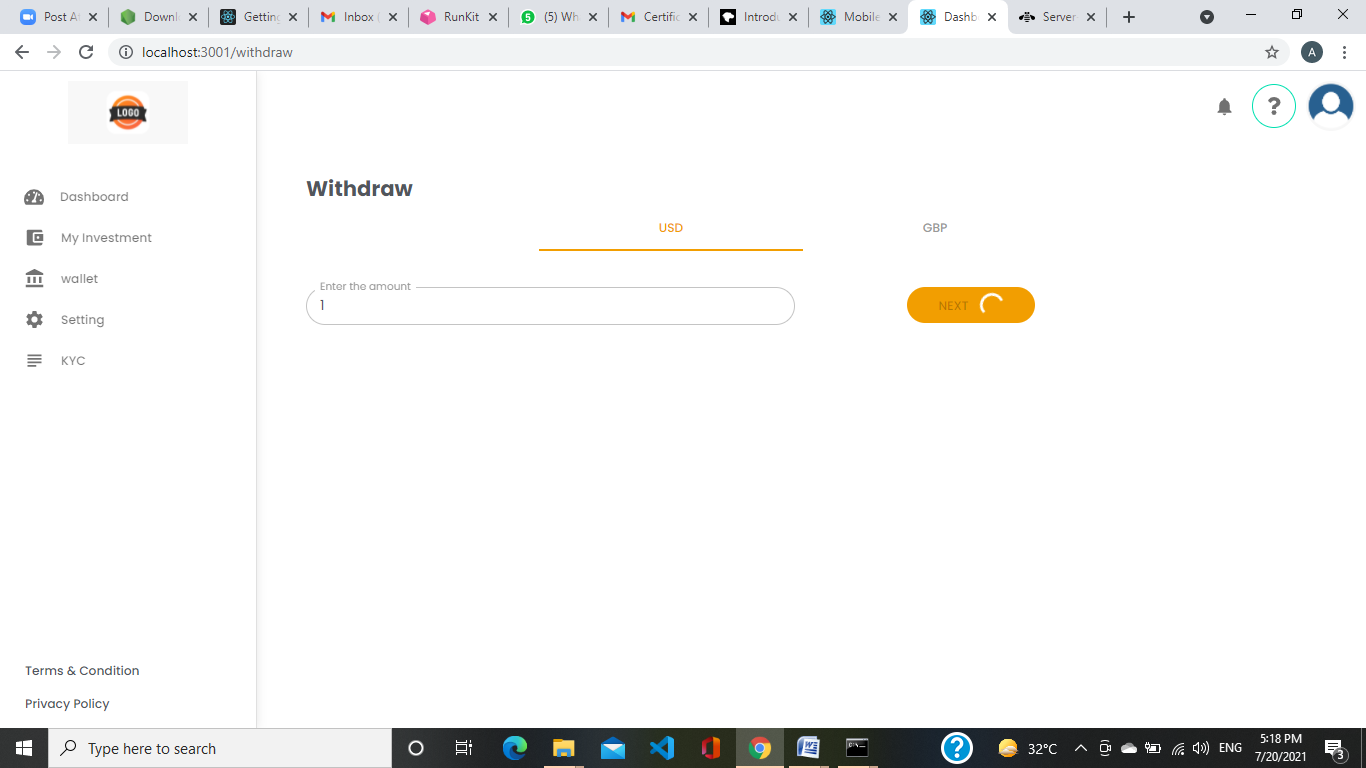


Fig. 4.10

### 4.11. Setting Screen:-

This is setting screen where user can on/off notification.

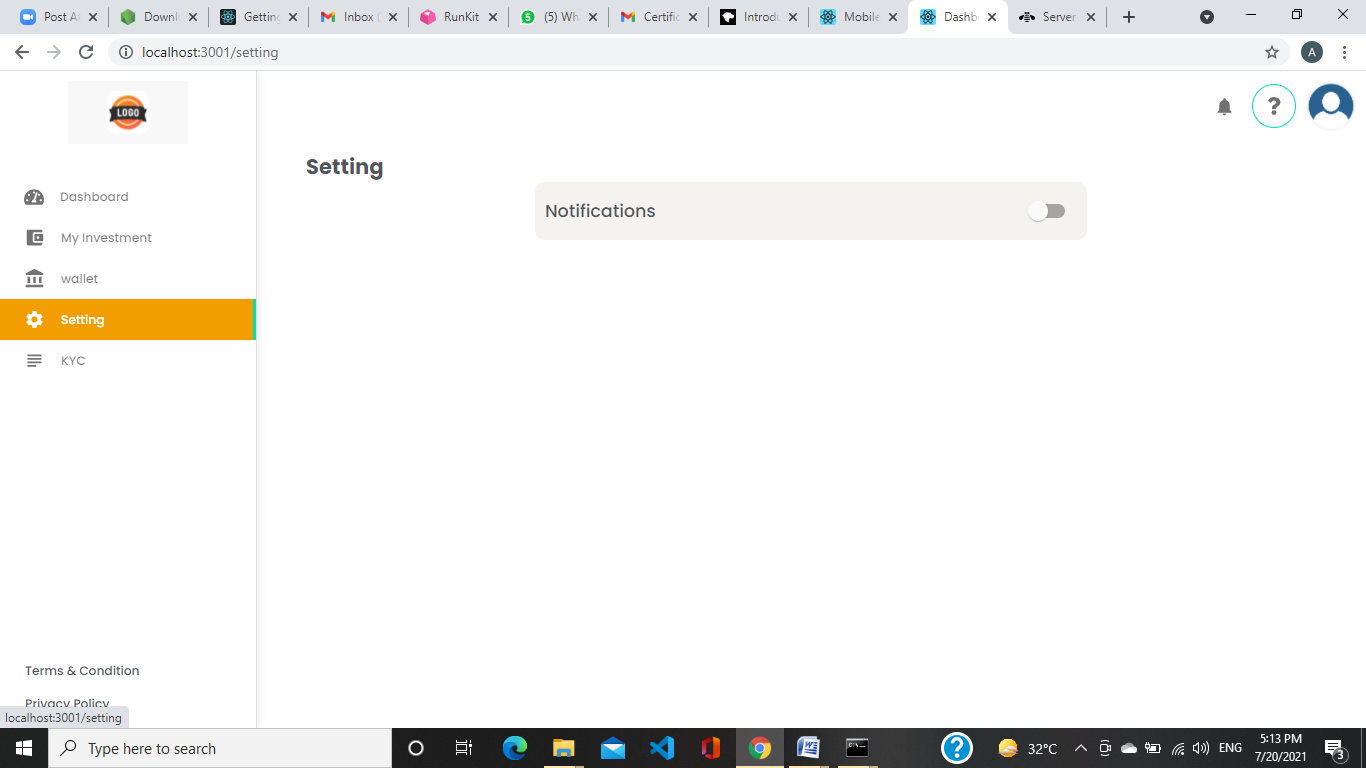


Fig. 4.11

### 4.12. KYC Screen:-

This is KYC screen where user can add or view the KYC user.

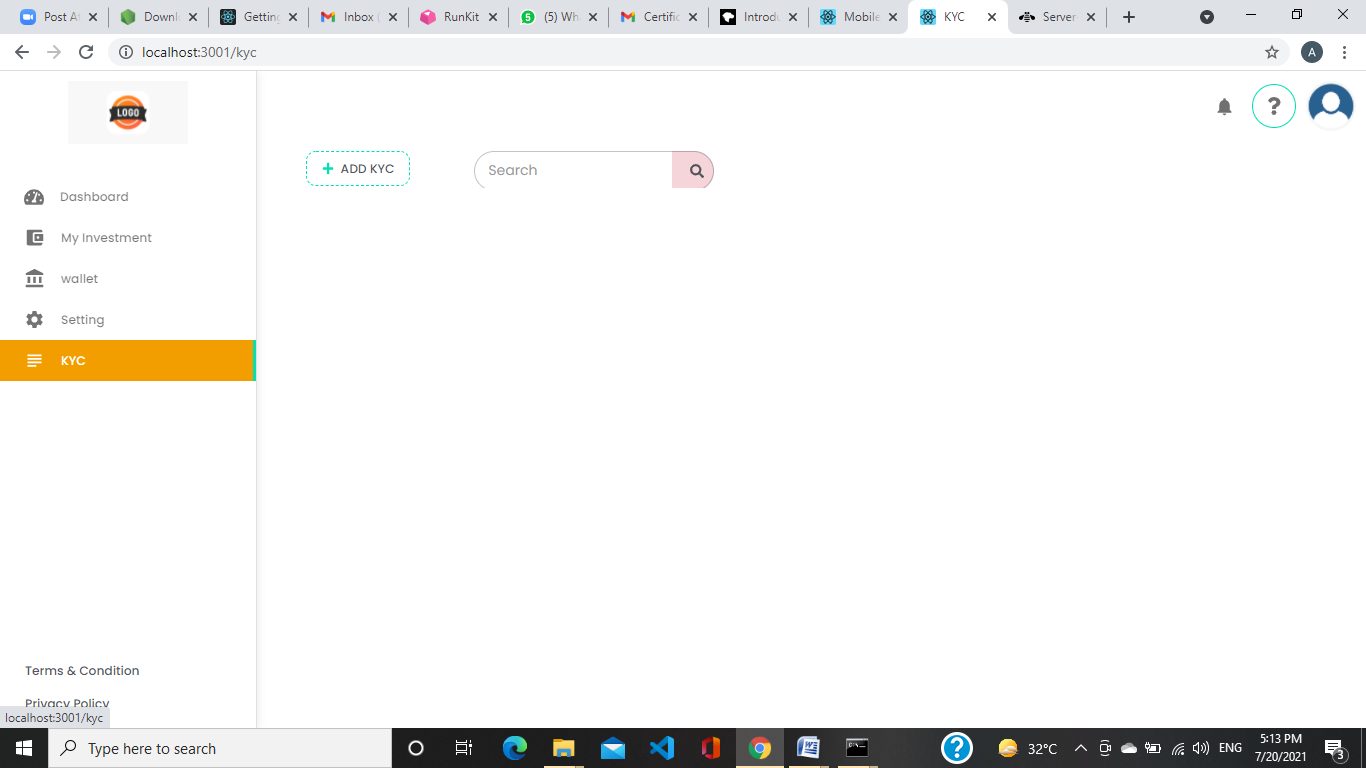


Fig- 4.12.1

This is KYC screen where user can verify their email.

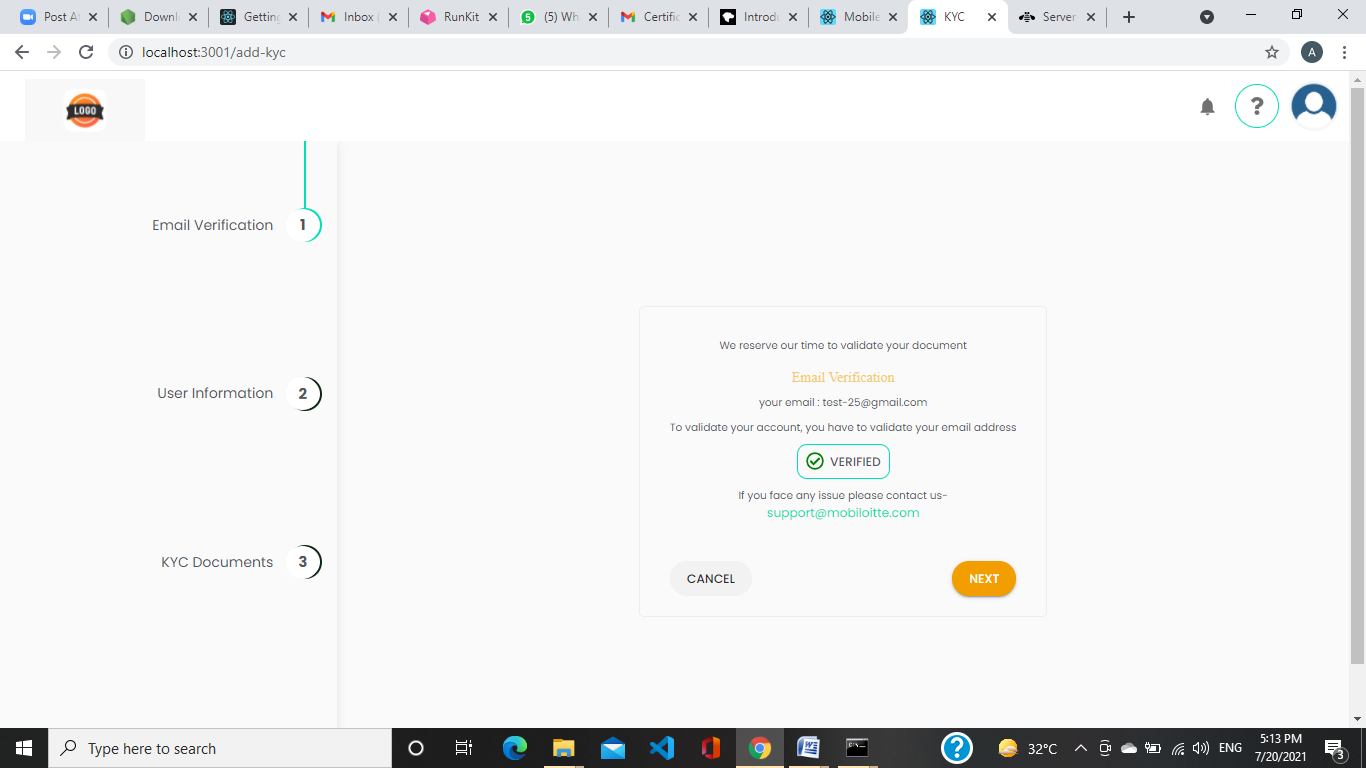


Fig- 4.12.2

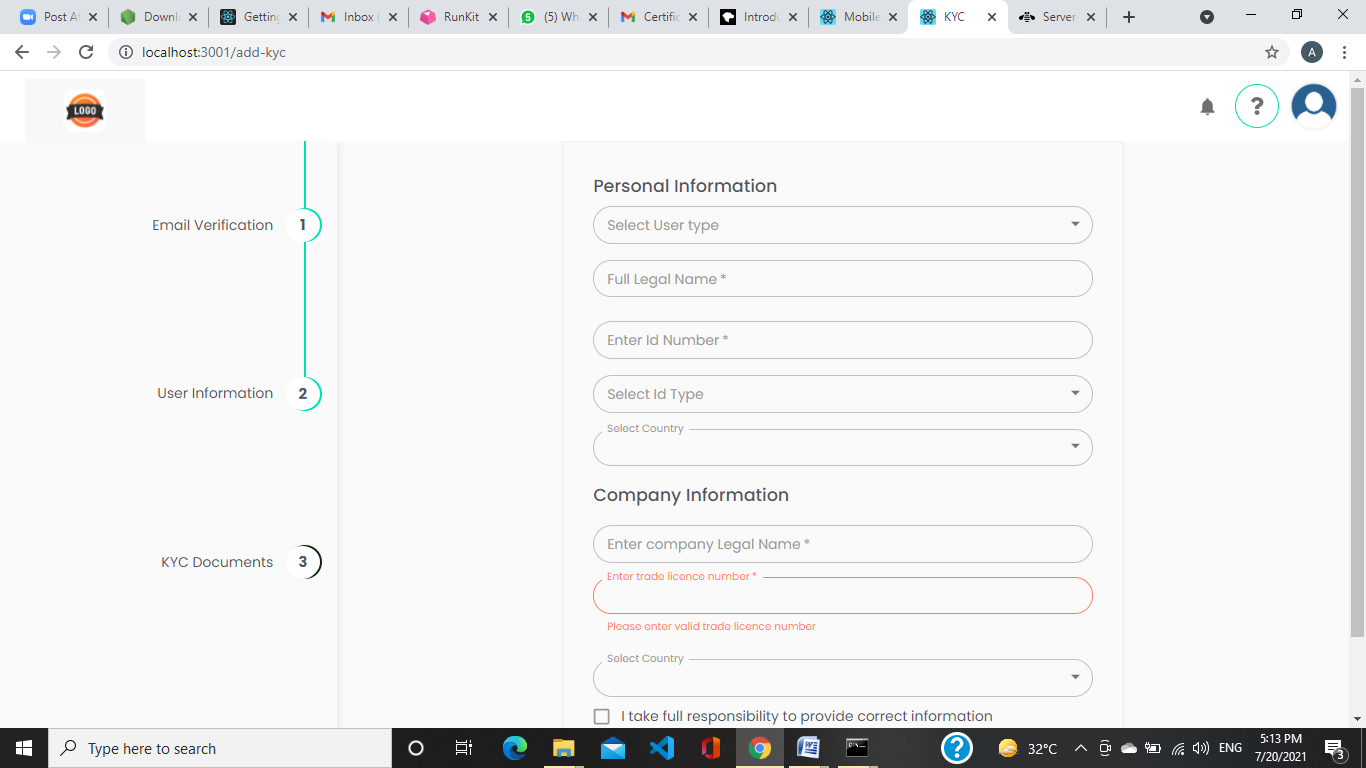


Fig- 4.12.3

This is KYC screen where user can enter all detail.

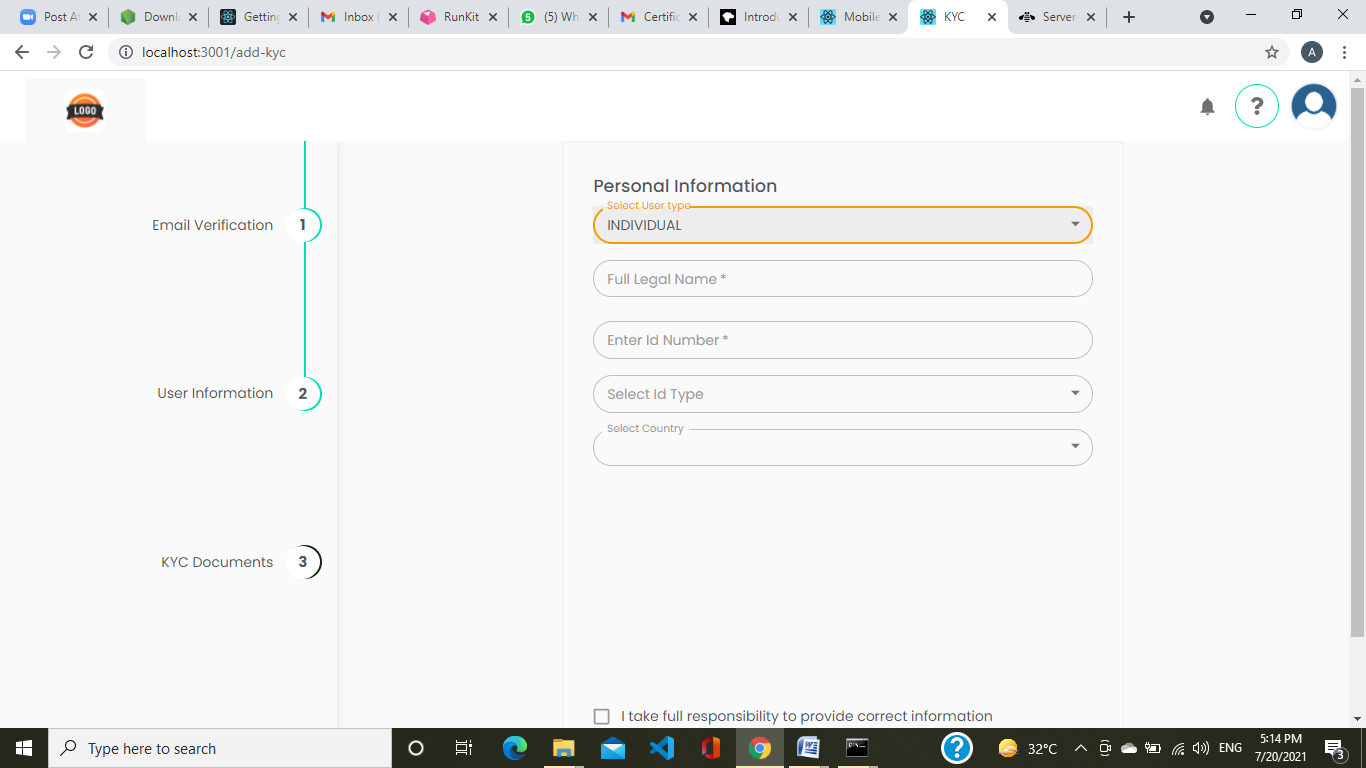


Fig- 4.12.4

This is KYC screen where user can select type of KYC Individual or Cooperate type.

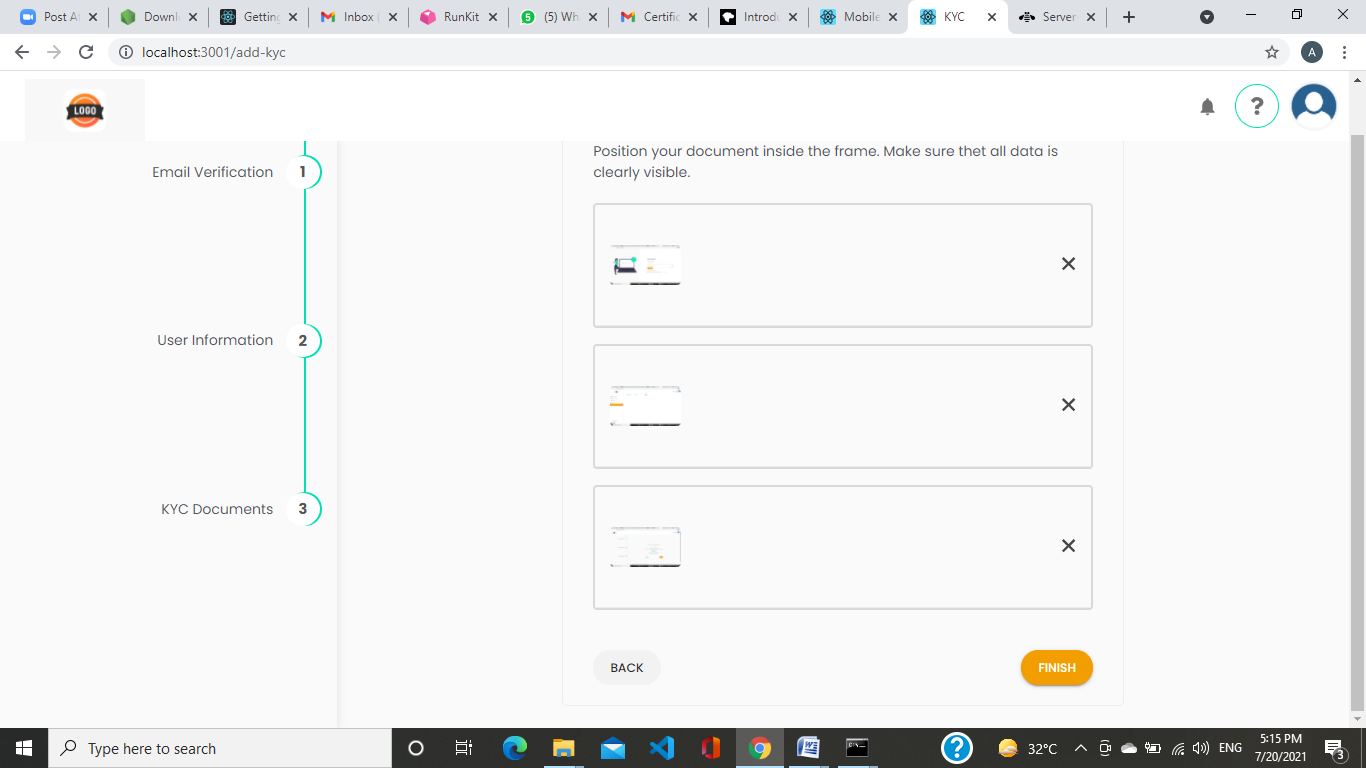


Fig- 4.12.5

This is KYC screen where user can upload their photo.

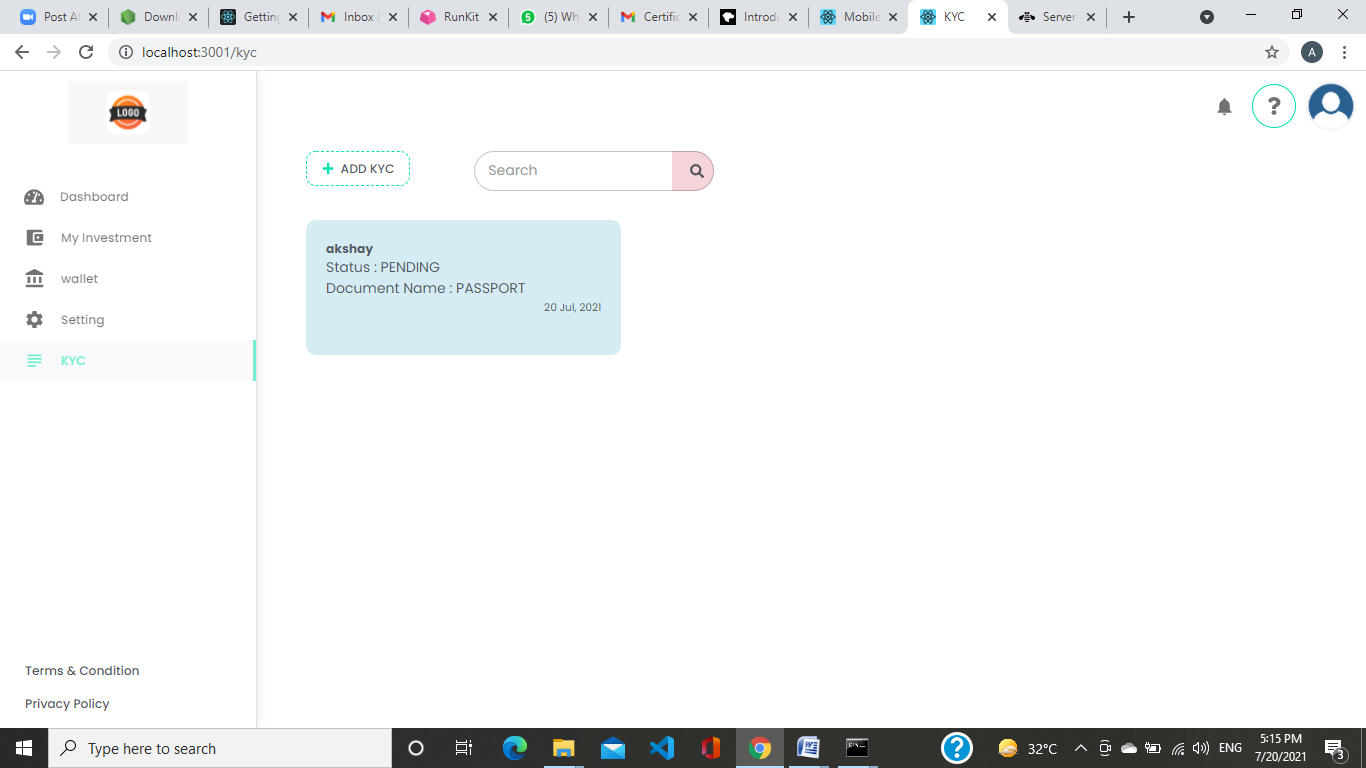


Fig- 4.12.6

This is show the KYC status to the user.

### 4.13. User Profile Screen:-

This is show the user profile and all details.

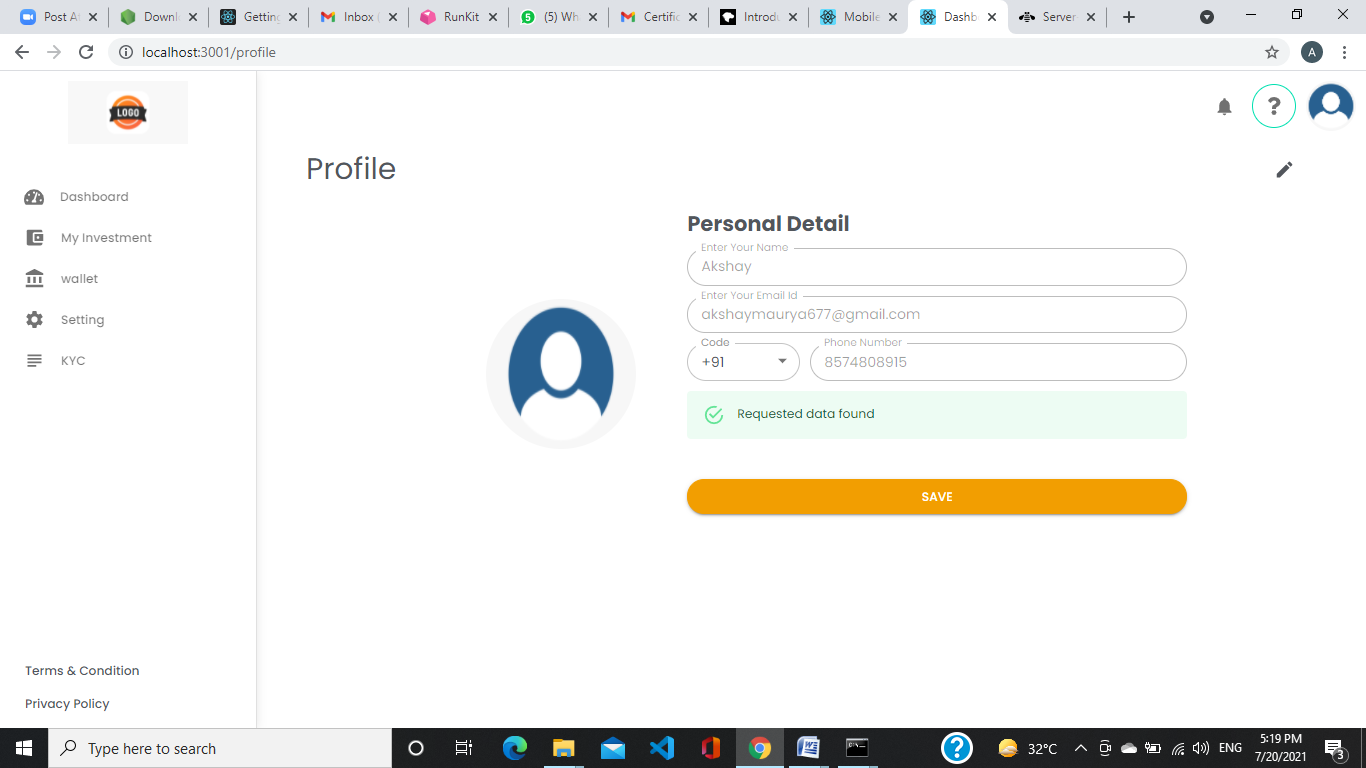


Fig. 4.13

Coding

import React, { useState,useContext } from "react";

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

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

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

import {

  Box,

  Button,

  Grid,

  TextField,

  Typography,

  FormControlLabel,

  Checkbox,

  FormControl,

  OutlinedInput,

  IconButton,

  InputAdornment,

} from "@material-ui/core";

import axios from "axios";

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

import {

  isValidPassword,

  isValidContact,

} from "../../../Validation/Validation";

import { AuthContext } from "src/context/Auth";

import ApiConfig from "src/config/APIConfig";

function Login(props) {

  // const history = useHistory();

  // const [number, setNumber] = useState(false);

  const [pass, setPass] = useState(false);

  const [values, setValues] = React.useState({

    password: "",

    showPassword: false,

  });

  const handleChange = (event) => {

    console.log("err", event);

    setValues({ ...values, [event.target.name]: event.target.value });

  };

  const handleClickShowPassword = () => {

    setValues({ ...values, showPassword: !values.showPassword });

  };

  const handleMouseDownPassword = (event) => {

    event.preventDefault();

  };

  const history = useHistory();

  const [phoneNumber, setPhoneNumber] = useState("");

const [password, setPassword]=useState("");

const [iserror, setIserror] = useState(false);

const [alertMsg, setAlertMsg] = useState("");

const [isSuccess, setIsSuccess] = useState(false);

const [isUpdating, setIsUpdating] = useState(false);

  const [number, setNumber] = useState(false);

  const [name, setName] = useState(true);

  const [err, setErr] = useState(false);

  const [isSubmit, setIsSubmit] = useState(false);

  const [formData, setFormData] = useState({

    // mobileNumber: "",

    number: "",

    password: "",

  });

  const \_onInputChange = (e) => {

    const name = e.target.name;

    const value = e.target.value;

    const temp = { ...formData, [name]: value };

    setFormData(temp);

  };

  const auth = useContext(AuthContext);

  const submitHandler = async () => {

    setIsSubmit(true);

    if (phoneNumber !== "" && password !== "") {

      setIsUpdating(true);

      await axios.post(ApiConfig.login,{

          phoneNumber: phoneNumber,

          password: password,

        })

        .then(async (response) => {

          console.log("response", response);

          if (response.data.responseCode !== 200) {

            setIserror(true);

            setIsSuccess(false);

            setErr("Your phone number or password is incorrect");

            // setAlertMsg(response.data.message);

          } else {

            setIserror(false);

            setErr("");

            setIsSuccess(true);

            auth.userLogIn(true, response.data.result.token);

            history.push("/dashboard");

            setIsUpdating(false);

          }

        })

        .catch((err) => {

          setIsUpdating(false);

          console.log(err);

        });

    }

  };

  return (

    <Box

      height="100%"

      display="flex"

      justifyContent="center"

      alignItems="center"

    >

      <Box display="flex" justifyContent="center">

        <Box padding={5} borderRadius={8} boxShadow={10} width="30%">

          <Grid container spacing={0}>

            <Grid item lg={12} md={12} sm={12} xs={12}>

              <Box display="flex" justifyContent="flex-start">

                <Typography variant="h2">Hello! let's get started</Typography>

              </Box>

              <Box display="flex" justifyContent="flex-start">

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

              </Box>

            </Grid>

            <Grid item lg={12} md={12} sm={12} xs={12}>

              <Box>

              <TextField

                type="number"

                placeholder="Enter Mobile Number"

                variant="outlined"

                size="small"

                // inputProps={{ style: { height: 14 } }}

                // onChange={mobileHandler}

                style={{ width: "100%", marginTop: 29,color:"red" }}

                name="phoneNumber"

                size="small"

                value={phoneNumber}

                onChange={(e) => setPhoneNumber(e.target.value)}

                error={phoneNumber !== "" && !isValidContact(phoneNumber)}

              ></TextField>

             {phoneNumber !== "" &&

                  !isValidContact(phoneNumber) && (

                    <span

                    style={{

                      variant: "subtitle2",

                      color: "red",

                      marginBottom: "10px",

                    }}

                    >

                     Please enter valid mobile number

                    </span>

                  )}

              </Box>

            </Grid>

            <Grid item lg={12} md={12} sm={12} xs={12}>

              <FormControl

                style={{ width: "100%", marginTop: 15 }}

                size="small"

                variant="outlined"

                size="small"

              >

                <OutlinedInput

                  placeholder="Enter Password"

                  id="outlined-adornment-password"

                  type={values.showPassword ? "text" : "password"}

                  value={values.password}

                  // inputProps={{ style: { height: 14 } }}

                  // onChange={(e)=>{handleChange(e);passHandler(e)}}

                  // onChange={passHandler}

                  endAdornment={

                    <InputAdornment position="end">

                      <IconButton

                        aria-label="toggle password visibility"

                        onClick={handleClickShowPassword}

                        onMouseDown={handleMouseDownPassword}

                        edge="end"

                      >

                        {values.showPassword ? (

                          <Visibility />

                        ) : (

                          <VisibilityOff />

                        )}

                      </IconButton>

                    </InputAdornment>

                  }

                  name="password"

                  value={password}

                  onChange={(e) => setPassword(e.target.value)}

                  error={password !== "" && !isValidPassword(password)}

                />

                {password !== "" &&

                  !isValidPassword(password) && (

                    <span

                    style={{

                      variant: "subtitle2",

                      color: "red",

                      marginBottom: "10px",

                    }}

                    >

                     At least one uppercase letter, one lowercase letter, one number and one special character

                    </span>

                  )}

              </FormControl>

            </Grid>

            <Grid

              container

              spacing={0}

              style={{ marginTop: 8, display: "flex", alignItems: "center" }}

            >

              <Grid item lg={6} md={6} sm={12} xs={12}>

                <Box

                  display="flex"

                  justifyContent="flex-start"

                  alignItems="center"

                >

                  <FormControlLabel

                  style={{marginRight:0}}

                    value="end"

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

                  />

                  <Typography>Remember me</Typography>

                </Box>

              </Grid>

              <Grid item lg={6} md={6} sm={12} xs={12}>

                <Box

                  display="flex"

                  justifyContent="flex-end"

                  alignItems="center"

                >

                  <Link onClick={() => history.push("/forgotPassword")}>

                    <Typography>Forgot Password?</Typography>

                  </Link>

                </Box>

              </Grid>

            </Grid>

            <Grid item lg={12} md={12} sm={12} xs={12}>

              <Box mt={1.5}>

                <Button

                  variant="contained"

                  style={{

                    width: "100%",

                    backgroundColor: "#252d47",

                    color: "white",

                  }}

                  // onClick={() => history.push("/dashboard")}

                  onClick={submitHandler}

                >

                  LOGIN

                </Button>

                <Typography >{err}</Typography>

              </Box>

            </Grid>

          </Grid>

        </Box>

      </Box>

    </Box>

  );

}

export default Login;

import React, { useState, useEffect } from "react";

import {

  Box,

  Button,

  Grid,

  TextField,

  Typography,

  makeStyles,

} from "@material-ui/core";

import Alert from "@material-ui/lab/Alert";

import {

  isValidPassword,

  isValidContact,

} from "../../../Validation/Validation";

import axios from "axios";

import ApiConfig from "src/config/APIConfig";

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

// import {

//   isValidPassword,

//   isValidContact,

// } from "../../../Validation/Validation";

const useStyles = makeStyles((theme) => ({

  root: { height: 435 },

  otp: {

    width: 30,

    height: 35,

    textAlign: "center",

    marginLeft: 10,

    border: "none",

    backgroundColor: "#9e9e9e",

    fontSize: 20,

  },

  text: {

    fontFamily: ["Poppins", "Helvetica", "Arial", "sans-serif"],

    fontSize: 14,

    marginLeft: 0,

  },

}));

function ForgotPassword(props) {

  const classes = useStyles();

  const [otp, setOtp] = useState(new Array(4).fill(""));

  const handleChanges = (element, index) => {

    setOtp([...otp.map((d, idx) => (idx === index ? element.value : d))]);

    //focus next element

    if (element.nextSibling) {

      element.nextSibling.focus();

    }

  };

  const history = useHistory();

  const [phoneNumber, setPhoneNumber] = useState(false);

  // const [otp , setOtp] = useState(false);

  const [oldPassword, setOldPassword] = useState("");

  // const [newPassword, setNewPassword] = useState("");

  const [\_id, setNew\_id] = useState("");

  const [isHide, setIsHide] = React.useState(true);

  // const [confirmPassword, setConfirmPassword] = useState("");

  const [iserror, setIserror] = useState(false);

  const [newpass, setNewpass] = useState("");

  const [confirmpass, setConfirmpass] = useState("");

  const [alertMsg, setAlertMsg] = useState("");

  const [isUpdating, setIsUpdating] = useState(false);

  const [err, setErr] = useState("");

  const [isSuccess, setIsSuccess] = useState(false);

  const [isSubmit, setIsSubmit] = useState(false);

  const [id, setId] = useState();

  const [values, setValues] = React.useState({

    password: "",

    showPassword: false,

  });

  const handleChange = (event) => {

    console.log("err", event);

    setValues({ ...values, [event.target.name]: event.target.value });

  };

  const handleClickShowPassword = () => {

    setValues({ ...values, showPassword: !values.showPassword });

  };

  const handleMouseDownPassword = (event) => {

    event.preventDefault();

  };

  function mobileHandler(e) {

    const regex = /^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$/;

    let phonNumber = "";

    if (e && e.target.value && regex.test(e.target.value)) {

      phonNumber = e.target.value;

    } else {

      phonNumber = e.target.value;

      setPhoneNumber(phonNumber);

    }

    console.warn("Enter valid phone number");

  }

  const newPassword = (e) => {

    setNewpass(e.target.value);

  };

  const confirmPassword = (e) => {

    setConfirmpass(e.target.value);

  };

  // function resetPassword(){

  const resetPassword = async () => {

    setIsSubmit(true);

    if (!isValidPassword(newpass) && !isValidPassword(confirmpass)) {

      alert("password mismatch");

    } else if (newPassword !== "" && confirmPassword !== "") {

      try {

        setIsUpdating(true);

        const response = await axios.put(

          `${ApiConfig.resetPassword}/${id}`,

          {

            newPassword: newpass,

            confirmPassword: confirmpass,

          },

          {

            headers: {

              token: ` ${accessToken}`,

            },

          }

        );

        console.log(response);

        if (response.data.responseCode !== 200) {

          setIserror(true);

          setIsSuccess(false);

          setAlertMsg(response.data.responseMessage);

        } else {

          setIsSuccess(true);

          setIserror(false);

          // history.push("/otp",email);

          setAlertMsg(response.data.responseMessage);

        }

        setIsUpdating(false);

      } catch (err) {

        console.log("ERROR", err);

        setIsUpdating(false);

      }

    }

  };

  const verifyOtp = async () => {

    setIsSubmit(true);

    if (otp !== "") {

      try {

        setIsUpdating(true);

        const vari = otp.toString();

        const response = await axios.post(

          ApiConfig.verifyOtp,

          {

            phoneNumber: phoneNumber,

            otp: parseInt(vari.replace(/,/g, "")),

          },

          {

            headers: {

              token: ` ${accessToken}`,

            },

          }

        );

        console.log(response);

        if (response.data.responseCode !== 200) {

          setIserror(true);

          setIsSuccess(false);

          setAlertMsg(response.data.responseMessage);

        } else {

          setIsSuccess(true);

          setIserror(false);

          setIsHide(false);

          setAlertMsg(response.data.responseMessage);

        }

        setIsUpdating(false);

      } catch (err) {

        console.log("ERROR", err);

        setIsUpdating(false);

      }

    }

  };

  const accessToken = window.localStorage.getItem("creatturAccessToken");

  console.log("token", accessToken);

  const submitHandler = async () => {

    setIsSubmit(true);

    if (phoneNumber !== "") {

      // saveFormData(formData);

      try {

        setIsUpdating(true);

        const accessToken = window.localStorage.getItem("creatturAccessToken");

        const response = await axios.post(

          ApiConfig.otp,

          { phoneNumber: phoneNumber },

          {

            headers: {

              token: ` ${accessToken}`,

            },

          }

        );

        console.log(response);

        if (response.data.responseCode !== 200) {

          setIserror(true);

          setIsSuccess(false);

          setAlertMsg(response.data.responseMessage);

        } else {

          setIsSuccess(true);

          setIserror(false);

          setId(response.data.result.\_id);

          // history.push("/otp",email);

          history.push({

            // pathname: '/otp',

            state: { phoneNumber: phoneNumber },

          });

          setAlertMsg(response.data.responseMessage);

        }

        setIsUpdating(false);

      } catch (err) {

        console.log("ERROR", err);

        setIsUpdating(false);

      }

    }

  };

  return (

    <Box

      height="100%"

      display="flex"

      justifyContent="center"

      alignItems="center"

    >

      <Box display="flex" justifyContent="center">

        <Box padding={5} borderRadius={8} boxShadow={10} width="35%">

          <Grid container spacing={0}>

            <Grid item lg={12} md={12} sm={12} xs={12}>

              <Box display="flex" justifyContent="flex-start">

                <Typography variant="h2">Forgot Password</Typography>

              </Box>

            </Grid>

            <Grid

              item

              lg={12}

              md={12}

              sm={12}

              xs={12}

              style={{ display: "flex" }}

            >

              <Box style={{ width: "75%" }}>

                <TextField

                  placeholder="Enter Mobile Number"

                  variant="outlined"

                  size="small"

                  onChange={(e) => setPhoneNumber(e.target.value)}

                  name="phoneNumber"

                  style={{ width: "100%", marginTop: 2 }}

                  error={phoneNumber !== "" && !isValidContact(phoneNumber)}

                />

                {phoneNumber !== "" && !isValidContact(phoneNumber) && (

                  <span

                    style={{

                      variant: "subtitle2",

                      color: "red",

                      marginBottom: "10px",

                    }}

                  >

                    Please enter valid mobile number

                  </span>

                )}

              </Box>

              <Box>

                <Button

                  variant="contained"

                  style={{

                    width: "100%",

                    backgroundColor: "#252d47",

                    color: "white",

                    marginLeft: 10,

                    marginTop: 5,

                  }}

                  onClick={submitHandler}

                  // onClick={() => history.push("/login")}

                >

                  SEND OTP

                </Button>

              </Box>

              {/\* </Box> \*/}

            </Grid>

            <Box mb="2%" mt={0.5}>

              <Typography className={classes.text}>

                Please enter the 4 digit OTP you received on your registered

                mobile number.

              </Typography>

            </Box>

            {isHide && (

              <Box

                style={{

                  marginTop: 5,

                  width: "100%",

                  // display: "flex",

                  // justifyContent: "space-between",

                }}

              >

                <Grid

                  container

                  spacing={0}

                  style={{

                    marginTop: 5,

                    width: "100%",

                    display: "flex",

                    justifyContent: "space-between",

                    alignItems: "center",

                  }}

                >

                  <Box>

                    <div style={{ textAlign: "center", marginBottom: "1.5%" }}>

                      {otp.map((data, index) => {

                        return (

                          <input

                            type="text"

                            name="OTP"

                            maxLength="1"

                            className={classes.otp}

                            key={index}

                            value={data}

                            onChange={(e) => handleChanges(e.target, index)}

                          />

                        );

                      })}

                    </div>

                  </Box>

                  <Box

                    style={{

                      display: "flex",

                      justifyContent: "flex-end",

                    }}

                  >

                    <Typography

                      style={{ marginBottom: "2%" }}

                      onClick={submitHandler}

                    >

                      <Link className={classes.link}>Resend OTP</Link>

                    </Typography>

                  </Box>

                </Grid>

                <Grid item lg={12} md={12} sm={12} xs={12}>

                  <Box textAlign="center">

                    <Button

                      variant="contained"

                      style={{

                        width: "50%",

                        backgroundColor: "#252d47",

                        color: "white",

                        marginTop: 30,

                        alignItems: "center",

                      }}

                      onClick={verifyOtp}

                    >

                      VERIFY

                    </Button>

                  </Box>

                </Grid>

              </Box>

            )}

            <Grid

              container

              spacing={0}

              style={{ marginTop: 0, display: "flex", alignItems: "center" }}

            >

              <Grid item lg={12} md={12} sm={12} xs={12}>

                <Box>

                  <TextField

                    label="Enter New Password :"

                    type="password"

                    variant="outlined"

                    required

                    name="newpass"

                    value={newpass}

                    fullWidth

                    // onChange={(e) => setNewpass(e.target.value)}

                    size="small"

                    inputProps={{ maxLength: 60 }}

                    style={{ marginTop: "10px" }}

                    onChange={newPassword}

                    error={newpass !== "" && !isValidPassword(newpass)}

                  />

                  {newpass !== "" && !isValidPassword(newpass) && (

                    <span

                      style={{

                        variant: "subtitle2",

                        color: "red",

                        marginBottom: "10px",

                      }}

                    >

                      At least one uppercase letter, one lowercase letter, one

                      number and one special character

                    </span>

                  )}

                </Box>

              </Grid>

              <Grid item lg={12} md={12} sm={12} xs={12}>

                <Box>

                <TextField

                  label="Confirm New Password :"

                  type="password"

                  variant="outlined"

                  required

                  name="confirmpass"

                  value={confirmpass}

                  fullWidth

                  // onChange={(e) => checkValidation(e)}

                  size="small"

                  inputProps={{ maxLength: 60 }}

                  style={{ marginTop: "10px" }}

                  onChange={confirmPassword}

                  error={

                    confirmpass !== "" && newpass !== confirmpass

                    // !isValidPassword(confirmnewpass)

                  }

                />

                 {confirmpass !== "" && newpass !== confirmpass && (

                    <span

                      style={{

                        variant: "subtitle2",

                        color: "red",

                        marginBottom: "10px",

                      }}

                    >

                      New password and confirm password should match

                    </span>

                  )}

                </Box>

              </Grid>

              <Grid item>

                {iserror && (

                  <Box my={2}>

                    <Alert severity="error">{alertMsg}</Alert>

                  </Box>

                )}

                {isSuccess && (

                  <Box mb={2}>

                    <Alert severity="success">{alertMsg}</Alert>

                  </Box>

                )}

              </Grid>

            </Grid>

            <Grid item lg={12} md={12} sm={12} xs={12}>

              <Box mt={4} textAlign="center">

                <Button

                  variant="contained"

                  style={{

                    width: "50%",

                    backgroundColor: "#252d47",

                    color: "white",

                    marginTop: 30,

                    // textAlign:"center"

                  }}

                  onClick={resetPassword}

                >

                  RESET PASSWORD

                </Button>

              </Box>

            </Grid>

          </Grid>

        </Box>

      </Box>

    </Box>

  );

}

export default ForgotPassword;

import { Typography, Box, Grid } from "@material-ui/core";

import React,{useEffect, useState} from "react";

import Page from "src/component/Page";

import Divider from "@material-ui/core/Divider";

import AccountBalanceIcon from "@material-ui/icons/AccountBalance";

import BarChartIcon from "@material-ui/icons/BarChart";

import FileCopyIcon from "@material-ui/icons/FileCopy";

import AttachMoneyIcon from "@material-ui/icons/AttachMoney";

import LocalAtmIcon from "@material-ui/icons/LocalAtm";

import MoneyIcon from "@material-ui/icons/Money";

import HowToRegIcon from "@material-ui/icons/HowToReg";

import TransformIcon from "@material-ui/icons/Transform";

import Card from "./card";

// import { getCountries } from "react-phone-number-input";

import axios from "axios";

import ApiConfig from "src/config/APIConfig";

export default function (props) {

  const accessToken = window.localStorage.getItem("creatturAccessToken");

  const [result, setresult] = useState([])

  const cardData1 = [

    {

      title: "TOTAL KYC",

      amount: result.TotalKyc,

      // result.TotalKyc

      icon: <FileCopyIcon fontSize="large" />,

    },

    {

      title: "TOTAL INVESTMENT PRODUCT",

      amount: result.TotalInvestment,

      icon: <BarChartIcon fontSize="large" />,

    },

    {

      title: "TOTAL WITHDRAWAL REQUEST",

      amount: result.TotalWithdraw,

      // icon:<AttachMoneyIcon fontSize='large'/>

      icon: <BarChartIcon fontSize="large" />,

    },

  ];

  const cardData2 = [

    {

      title: "TOTAL TRANSACTION",

      amount: result.TotalTransaction,

      icon: <LocalAtmIcon fontSize="large" />,

    },

    {

      title: "TOTAL DEPOSIT BALANCE",

      amount: 85,

      icon: <MoneyIcon fontSize="large" />,

    },

    {

      title: "TOTAL REGISTERED USERS",

      amount: result.TotalUser,

      icon: <HowToRegIcon fontSize="large" />,

    },

  ];

  const cardData3 = [

    {

      title: "TOTAL WITHDRAWAL AMOUNT",

      amount: 85,

      icon: <AttachMoneyIcon fontSize="large" />,

    },

    {

      title: "PENDING WITHDRAWAL REVIEW ",

      amount: result.TotalPendingWithdrawResult,

      icon: <AccountBalanceIcon fontSize="large" />,

    },

    {

      title: "PENDING DEPOSIT REVIEW",

      amount: result.TotalPendingDeposit,

      icon: <TransformIcon fontSize="large" />,

    },

  ];

 const getCount=async()=>{

   await axios.get(ApiConfig.getcount,{

     headers:{

       token:accessToken,

     }

   }).then((response)=>{

    if(response.data.responseCode==200){

      // console.log(response.data.responseCode)

      // console.log(response.data.result)

      // setListData(response.data.result)

      // console.log("result",response.data.result)

      setresult(response.data.result)

    }

    else{

      console.log("data not found")

    }

 })}

 console.log("result",result)

 console.log("result",result.TotalKyc)

  useEffect(() => {

    getCount()

  }, [])

  return (

    <Page title="Dashboard">

      <Box p={2} pb={"80px"}>

        <Typography variant="h3" style={{ marginBottom: "10px" }}>

          <strong>DASHBOARD</strong>

        </Typography>

        <Divider />

        <Grid container spacing={3}>

          {cardData1.map((obj) => (

            <Grid item md={4} xs={12}>

              <Card {...obj} />

            </Grid>

          ))}

        </Grid>

        <Grid container spacing={3}>

          {cardData2.map((obj) => (

            <Grid item md={4} xs={12}>

              <Card {...obj} />

            </Grid>

          ))}

        </Grid>

        <Grid container spacing={3}>

          {cardData3.map((obj) => (

            <Grid item md={4} xs={12}>

              <Card {...obj} />

            </Grid>

          ))}

        </Grid>

      </Box>

    </Page>

  );

}

import { Typography, Box, makeStyles, } from "@material-ui/core";

import React from "react";

const useStyles = makeStyles((theme) => ({

    main: {

         backgroundColor: theme.palette.primary.main,

         display:'flex',

         flexDirection:'column',

         alignItems:'center',

         justifyContent:'center',

         height: '200px',

         borderRadius: '4px',

         color:'#ffff',

         WebkitTransform:' all .3s ease-in-out .3s',

         MozTransform: 'all .1s ease-in-out .3s',

         msTransform: 'all .1s ease-in-out .3s',

         OTransform: 'all .1s ease-in-out .3s',

         transition: 'all .1s ease-in-out .3s',

         '&:hover': {

            WebkitTransform: 'translateY(-9px)',

            MozTransform: 'translateY(-9px)',

            msTransform: 'translateY(-9px)',

            OTransform: 'translateY(-9px)',

            transform:' translateY(-9px)',

            WebkitTransform: 'all .3s ease-all .3s',

            MozTransform:' all .1s ease-in-out .3s',

            msTransform: 'all .1s ease-in-out .3s',

            OTransform: 'all .1s ease-in-out .3s',

            transition: 'all .1s ease-in-out .3s',

       },

    }

}));

const Card = ({title,amount,icon}) => {

    const classes = useStyles();

    return <Box mt={7} className={classes.main}  >

        {icon}

       <Typography variant='h4' style={{marginTop:'20px'}}>{title}</Typography>

       <Typography variant='h4'>{amount}</Typography>

    </Box>

}

export default Card;

import React, { useState,useEffect } from "react";

import SearchFilter from "../../../component/SearchFilter";

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

import {

  Container,

  Divider,

  Box,

  Paper,

  Typography,

  Button,

  Link,

  TableCell,

  TableContainer,

  TableHead,

  TableRow,

  TableBody,

  Table,

} from "@material-ui/core";

import Pagination from '@material-ui/lab/Pagination';

import VisibilityIcon from "@material-ui/icons/Visibility";

import Page from "src/component/Page";

import { makeStyles } from "@material-ui/core/styles";

import axios from "axios";

import ApiConfig from "src/config/APIConfig";

import { DataGrid } from "@material-ui/data-grid";

const useStyles = makeStyles({

  table: {

    minWidth: 320,

  },

  pdbt: {

    paddingBottom: 52,

  },

  button: {

    minWidth: "initial",

    padding: "6px",

    marginLeft: "7px",

  },

});

export default function (props) {

  const classes = useStyles();

  function createData(

    Sr\_No,

    LegalName,

    Document\_Id\_Type,

    Document\_Id\_Number,

    Country,

    Status,

    Actions

  ) {

    return {

      Sr\_No,

      LegalName,

    Document\_Id\_Type,

    Document\_Id\_Number,

    Country,

    Status,

      Actions,

    };

  }

  const rows = [

    createData(

      1,

      "Rose",

      "privet",

      737555148459,

      "India",

      "Approved",

    ),

    createData(

      2,

      "Rose",

      "privet",

      737555148459,

      "India",

      "Approved",

    ),

    createData(

      3,

      "Rose",

      "privet",

      737555148459,

      "India",

      "Approved",

    ),

    createData(

      4,

      "Rose",

      "privet",

      737555148459,

      "India",

      "Approved",

    ),

    createData(

      5,

      "Rose",

      "privet",

      737555148459,

      "India",

      "Approved",

    ),

  ];

  function download\_table\_as\_csv(table\_id, separator = ',') {

    // Select rows from table\_id

    var rows = document.querySelectorAll('table#' + table\_id + ' tr');

    // Construct csv

    var csv = [];

    for (var i = 0; i < rows.length; i++) {

        var row = [], cols = rows[i].querySelectorAll('td, th');

        for (var j = 0; j < cols.length; j++) {

            // Clean innertext to remove multiple spaces and jumpline (break csv)

            var data = cols[j].innerText.replace(/(\r\n|\n|\r)/gm, '').replace(/(\s\s)/gm, ' ')

            // Escape double-quote with double-double-quote (see https://stackoverflow.com/questions/17808511/properly-escape-a-double-quote-in-csv)

            data = data.replace(/"/g, '""');

            // Push escaped string

            row.push('"' + data + '"');

        }

        csv.push(row.join(separator));

    }

    var csv\_string = csv.join('\n');

    // Download it

    var filename = 'export\_' + table\_id + '\_' + new Date().toLocaleDateString() + '.csv';

    var link = document.createElement('a');

    link.style.display = 'none';

    link.setAttribute('target', '\_blank');

    link.setAttribute('href', 'data:text/csv;charset=utf-8,' + encodeURIComponent(csv\_string));

    link.setAttribute('download', filename);

    document.body.appendChild(link);

    link.click();

    document.body.removeChild(link);

  }

  const [selectedTab, setTab] = useState("individual");

  const tabChange = (event, tabName) => {

    setTab(tabName);

  };

  const [iserror, setIserror] = useState(false);

  const [alertMsg, setAlertMsg] = useState("");

  const [isSuccess, setIsSuccess] = useState(false);

  const [isUpdating, setIsUpdating] = useState(false);

  const [kycData, setKycDaTA] = useState([]);

  const accessToken = window.localStorage.getItem("creatturAccessToken");

  useEffect(() => {

    // setIsLoading(true);

    getlist();

  }, []);

  const getlist=async () =>{

    await axios.post(ApiConfig.getAllKycDetail,{}, {

      headers:{

       token:accessToken,

      }

     })

     .then((response) => {

      // console.log("data",response);

      if (response.data.responseCode !== 200) {

        setIserror(true);

        setIsSuccess(false);

        setAlertMsg(response.data.responseMessage);

      } else {

       setIsSuccess(true);

       setIserror(false);

      // setIsLoading(false);

      setAlertMsg(response.data.responseMessage);

      setIsUpdating(false);

     setKycDaTA(response.data.result.docs)

      }

    })

    .catch((response) => {

      setIsUpdating(false);

      console.log("response", response);

    });

};

  //  const columns = selectedTab === "individual" ? columns1 : columns2;

  // const rows = selectedTab === "individual" ? rows1 : rows2;

  return (

    <Container

      maxWidth="xl"

      style={{  paddingBottom: "100px" }}

    >

      <Page

        style={{ display: "flex", flexDirection: "column" }}

        title="KYC Management"

      >

        <Box py={3}>

          <Typography variant="h3" style={{ marginBottom: "8px" }}>

            <strong>KYC MANAGEMENT</strong>

          </Typography>

          <Divider />

          <Box py={5}>

            <SearchFilter

              exportProps={{ title: "Export CSV", onClick: download\_table\_as\_csv, name:'kyc\_list' }}

              showTabs

              searchProps="Search By Name"

              selectedTab={selectedTab}

              tabChange={tabChange}

            />

          </Box>

        </Box>

        <Box style={{ overflow: "auto" }}>

          {/\* <DataGrid

            className={classes.pdbt}

            rows={rows}

            columns={columns}

            pageSize={5}

            autoHeight

            style={{ minWidth: "1276px" }}

          />

        </Box> \*/}

        <TableContainer component={Paper}>

          <Table className={classes.table} aria-label="simple table" id='kyc\_list'>

            <TableHead>

              <TableRow>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                  align="left"

                >

                  Sr.No

                </TableCell>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                >

                  Legal Name

                </TableCell>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                  align="left"

                >

                  Document Id Type

                </TableCell>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                  align="left"

                >

                  Document Id Number

                </TableCell>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                  align="left"

                >

                  Country

                </TableCell>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                  align="left"

                >

                  Status

                </TableCell>

                <TableCell

                  style={{ color: "white", backgroundColor: "#252d47" }}

                  align="left"

                >

                  Action

                </TableCell>

              </TableRow>

            </TableHead>

            <TableBody>

              {kycData.map((row, i) => (

                <TableRow key={row.name}>

                  <TableCell component="th" scope="row">

                    {i+1}

                  </TableCell>

                  <TableCell align="left">{row.fullLegalName}</TableCell>

                  <TableCell align="left">{row.IdType}</TableCell>

                  <TableCell align="left">{row.idNumber}</TableCell>

                  <TableCell align="left">{row.country}</TableCell>

                  <TableCell align="left">{row.kycStatus}</TableCell>

                  <TableCell style={{ width: 5 }} align="left">

                    <Box display="flex">

                    <Link

                     to={{

                      pathname: "/viewKYC",

                      state: {

                        data: row.\_id,

                      },

                    }}

                    component={RouterLink}>

            <Button

              variant="contained"

              color="primary"

              className={classes.button}

            >

              <VisibilityIcon style={{ fontSize: "15px" }} />

            </Button>

          </Link>

                      </Box>

                  </TableCell>

                </TableRow>

              ))}

            </TableBody>

          </Table>

          {/\* <ul className={classes.ul}>

            {items.map(({ page, type, selected, ...item }, index) => {

              let children = null;

              if (type === "start-ellipsis" || type === "end-ellipsis") {

                children = "…";

              } else if (type === "page") {

                children = (

                  <button

                    type="button"

                    style={{ fontWeight: selected ? "bold" : undefined }}

                    {...item}

                  >

                    {page}

                  </button>

                );

              } else {

                children = (

                  <button type="button" {...item}>

                    {type}

                  </button>

                );

              }

              return <li key={index}>{children}</li>;

            })}

          </ul> \*/}

        </TableContainer>

        <Box display="flex" justifyContent="flex-end">

          <Pagination count={10} shape="rounded" />

          </Box>

        </Box>

      </Page>

    </Container>

  );

}

import React, { useState, useEffect } from "react";

import {

  Container,

  Grid,

  Box,

  Typography,

  Button,

  Card,

  CardHeader,

  CardContent,

  Dialog,

  TextField,

  makeStyles,

  Link,

} from "@material-ui/core";

import DialogActions from "@material-ui/core/DialogActions";

import DialogContent from "@material-ui/core/DialogContent";

import DialogContentText from "@material-ui/core/DialogContentText";

import PanFront from "../../../images/panFront.png";

import PanRear from "../../../images/panRear.png";

import Selfie from "../../../images/selfie.jpeg";

import Page from "src/component/Page";

import axios from "axios";

import ApiConfig from "src/config/APIConfig";

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

const useStyles = makeStyles((theme) => ({

  images: {

    WebkitTransform: " all .3s ease-in-out .2s",

    MozTransform: "all .1s ease-in-out .2s",

    msTransform: "all .1s ease-in-out .2s",

    OTransform: "all .1s ease-in-out .2s",

    transition: "all .1s ease-in-out .2s",

    "&:hover": {

      WebkitTransform: "scale(1.2)",

      MozTransform: "scale(1.2)",

      msTransform: "scale(1.2)",

      OTransform: "scale(1.2)",

      transform: " scale(1.2)",

      WebkitTransform: "all .3s ease-all .2s",

      MozTransform: " all .1s ease-in-out .2s",

      msTransform: "all .1s ease-in-out .2s",

      OTransform: "all .1s ease-in-out .2s",

      transition: "all .1s ease-in-out .2s",

    },

  },

}));

const Row = ({ field, value }) => (

  <Grid item container md={12}>

    <Grid item md={6}>

      <Box display="flex" justifyContent="space-between" pr={4}>

        <Typography variant="h3" style={{ fontSize: 18 }}>

          {field}

        </Typography>

        :

      </Box>

    </Grid>

    <Grid item md={6}>

      <Typography style={{ fontSize: 14 }} variant="body1">

        {value}

      </Typography>

    </Grid>

  </Grid>

);

const ViewKYC = (props) => {

  const classes = useStyles();

  const history = useHistory();

  const [isApprove, setApprove] = React.useState(false);

  const productId = props.location && props.location.state.data;

  const openAprove = () => {

    setApprove(true);

  };

  const closeApprove = () => {

    setApprove(false);

  };

  const [isReject, setReject] = React.useState(false);

  const openReject = () => {

    setReject(true);

  };

  const closeReject = () => {

    setReject(false);

  };

  const [iserror, setIserror] = useState(false);

  const [alertMsg, setAlertMsg] = useState("");

  const [isSuccess, setIsSuccess] = useState(false);

  const [isUpdating, setIsUpdating] = useState(false);

  const [kycDetails, setKycDetails] = useState({

    KYCId: "",

    KYCStatus: "",

    CreatedAt: "",

    Type: "",

    IdType: "",

    IdNumber: "",

    CompanyLegalName: "",

    TradeLicenceNumber: "",

    // LicenceIssueBy: "",

    Front: "",

    Rear: "",

    Selfie: "",

  });

  const accessToken = window.localStorage.getItem("creatturAccessToken");

  useEffect(() => {

    // setIsLoading(true);

    console.log(productId, accessToken);

    axios

      .get(`${ApiConfig.getPaticularKyc}/${productId} `, {

        headers: {

          token: accessToken,

        },

      })

      .then((response) => {

        if (response.data.responseCode !== 200) {

        } else {

          console.log(response.data.result);

          setKycDetails({

            ...kycDetails,

            KYCId: response.data.result.kycUserId,

            KYCStatus: response.data.result.kycStatus,

            CreatedAt: response.data.result.createdAt,

            Type: response.data.result.IdType,

            IdType: response.data.result.IdType,

            IdNumber: response.data.result.idNumber,

            CompanyLegalName: response.data.result.companyLegalName,

            TradeLicenceNumber: response.data.result.tradeLicNumber,

            // LicenceIssueBy: response.data.result.,

            Front: response.data.result.docFrontPage,

            Rear: response.data.result.docBackPage,

            Selfie: response.data.result.selfie,

          });

        }

      })

      .catch((response) => {

        // setIsUpdating(false);

        console.log("response", response);

      });

  }, []);

  const submitHandler = async () => {

    // setIsSubmit(true);

    // saveFormData(formData);

    let payload = {

      kycId: productId,

    };

    try {

      setIsUpdating(true);

      const accessToken = window.localStorage.getItem("creatturAccessToken");

      const response = await axios.post(ApiConfig.approveKycByAdmin, payload, {

        headers: {

          token: `${accessToken}`,

        },

      });

      console.log(response);

      if (response.data.responseCode !== 200) {

        setIserror(true);

        setIsSuccess(false);

        setAlertMsg(response.data.responseMessage);

      } else {

        setIsSuccess(true);

        setIserror(false);

        // history.push("/kyc");

        setAlertMsg(response.data.responseMessage);

      }

      setIsUpdating(false);

    } catch (err) {

      console.log("ERROR", err);

      setIsUpdating(false);

    }

  };

  const submitHandlerReject = async () => {

    // setIsSubmit(true);

    // saveFormData(formData);

    let payload = {

      kycId: productId,

    };

    try {

      setIsUpdating(true);

      const accessToken = window.localStorage.getItem("creatturAccessToken");

      const response = await axios.post(ApiConfig.rejectedKyc, payload, {

        headers: {

          token: `${accessToken}`,

        },

      });

      console.log(response);

      if (response.data.responseCode !== 200) {

        setIserror(true);

        setIsSuccess(false);

        setAlertMsg(response.data.responseMessage);

      } else {

        setIsSuccess(true);

        setIserror(false);

        history.push("/kyc");

        setAlertMsg(response.data.responseMessage);

      }

      setIsUpdating(false);

    } catch (err) {

      console.log("ERROR", err);

      setIsUpdating(false);

    }

  };

  return (

    <>

      <Page

        style={{ display: "flex", flexDirection: "column" }}

        title="View KYC"

      >

        <Container

          maxWidth="xl"

          style={{ backgroundColor: "#fafafa", paddingBottom: "50px" }}

        >

          <Box py={3}>

            <Typography variant="h2">View KYC</Typography>

            <Container maxWidth="lg">

              <Card>

                <CardHeader

                  title="KYC DETAILS"

                  style={{

                    borderBottom: "1px solid rgba(0,0,0,0.125)",

                    backgroundColor: "rgba(0,0,0,.03)",

                  }}

                />

                <CardContent>

                  <Grid container>

                    <Grid container item md={6} direction="column">

                      <Row field="KYC ID" value={kycDetails.KYCId} />

                      <Row field="KYC Status" value={kycDetails.KYCStatus} />

                      <Row field="Created At" value={kycDetails.CreatedAt} />

                      {/\* <Row field="Type" value="eKYC" /> \*/}

                      <Row field="Id Type" value={kycDetails.IdType} />

                      <Row field="Id Number" value={kycDetails.IdNumber} />

                    </Grid>

                    <Grid container item md={6}>

                      <Row field="Company Legal Name" value={kycDetails.CompanyLegalName} />

                      <Row

                        field="Trade Licence Number"

                        value={kycDetails.TradeLicenceNumber}

                      />

                      {/\* <Row

                        field="Licence Issue By"

                        value="Value Added Tax (VAT) registration certificate"

                      /> \*/}

                    </Grid>

                  </Grid>

                </CardContent>

              </Card>

              <Box pt={4}>

                <Grid

                  container

                  className="kyc images"

                  justify="space-around"

                  alignItems="center"

                >

                  <Grid item md={4}>

                    {" "}

                    <img

                      src={kycDetails.Front}

                      alt="front"

                      // sizes="small"

                      width="260px"

                      className={classes.images}

                    />{" "}

                    <Typography variant="h3" style={{marginTop:7}}>Front</Typography>

                  </Grid>

                  <Grid item md={4}>

                    <img

                      src={kycDetails.Rear}

                      alt="front"

                      // sizes="small"

                      width="260px"

                      className={classes.images}

                    />

                    <Typography variant="h3" style={{marginTop:7}}>Rear</Typography>

                  </Grid>

                  <Grid item md={4}>

                    <img

                      src={kycDetails.Selfie}

                      alt="front"

                      width="260px"

                      className={classes.images}

                    />

                    <Typography variant="h3" style={{marginTop:7}}>Selfie</Typography>

                  </Grid>

                </Grid>

              </Box>

              <Box

                className="buttons"

                display="flex"

                justifyContent="center"

                mt={5}

              >

                <Button

                  variant="contained"

                  color="primary"

                  size="large"

                  onClick={openAprove}

                >

                  Approve{" "}

                </Button>

                <Button

                  variant="contained"

                  color="primary"

                  size="large"

                  style={{ marginLeft: "10px", marginRight: "10px" }}

                  onClick={openReject}

                >

                  Reject

                </Button>

                <Link href="/kyc">

                  <Button variant="contained" color="primary" size="large">

                    Back

                  </Button>

                </Link>

              </Box>

              <Dialog

                open={isApprove}

                onClose={closeApprove}

                aria-labelledby="alert-dialog-title"

                aria-describedby="alert-dialog-description"

              >

                <DialogContent>

                  <DialogContentText

                    id="alert-dialog-description"

                    style={{ color: "#404040" }}

                  >

                    Are you sure you want to approve this document?

                  </DialogContentText>

                </DialogContent>

                <DialogActions>

                  <Button color="primary" onClick={submitHandler}>

                    Yes

                  </Button>

                  <Button onClick={closeApprove} color="primary" autoFocus>

                    No

                  </Button>

                </DialogActions>

              </Dialog>

              <Dialog

                open={isReject}

                onClose={closeReject}

                aria-labelledby="alert-dialog-title"

                aria-describedby="alert-dialog-description"

              >

                <DialogContent>

                  <DialogContentText

                    id="alert-dialog-description"

                    style={{ color: "#404040" }}

                  >

                    Are you sure you want to reject this document?

                  </DialogContentText>

                  <TextField

                    fullWidth

                    placeholder="Specify reason fo rejection"

                  />

                </DialogContent>

                <DialogActions>

                  <Button color="primary" onClick={submitHandlerReject}>

                    Yes

                  </Button>

                  <Button onClick={closeReject} color="primary" autoFocus>

                    No

                  </Button>

                </DialogActions>

              </Dialog>

            </Container>

          </Box>

        </Container>

      </Page>

    </>

  );

};

export default ViewKYC;

import React, { useState,useEffect,useContext } from "react";

import {

  Container,

  Divider,

  Box,

  Link,

  Typography,

  Button,

  Grid,

  Card,

  CardContent,

  TextField,

  TableCell,

  TableContainer,

  TableHead,

  TableRow,

  TableBody,

  Table,

  Paper,

  FormControl,

  OutlinedInput,

  InputAdornment,

} from "@material-ui/core";

import VisibilityIcon from "@material-ui/icons/Visibility";

import SearchIcon from "@material-ui/icons/Search";

import { isValidMetric } from "../../../Validation/Validation";

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

import axios from "axios";

import { AuthContext } from "src/context/Auth";

import Pagination from "@material-ui/lab/Pagination";

import ApiConfig from "src/config/APIConfig";

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

import DateFnsUtils from "@date-io/date-fns";

import {

  MuiPickersUtilsProvider,

  KeyboardDatePicker,

} from "@material-ui/pickers";

import Modal from "react-modal";

import Tabs from "@material-ui/core/Tabs";

import Tab from "@material-ui/core/Tab";

import Page from "src/component/Page";

import { makeStyles } from "@material-ui/core/styles";

import useIsMountedRef from "src/component/useIsMountedRef";

const useStyles = makeStyles({

  table: {

    minWidth: 320,

    marginBottom:100

  },

  pdbt: {

    paddingBottom: 52,

  },

  button: {

    minWidth: "initial",

    padding: "6px",

    marginLeft: "7px",

  },

});

const accessToken = window.localStorage.getItem("creatturAccessToken");

export default function (props) {

  const classes = useStyles();

  const [addNew, setAddNew] = useState(false);

  const [userData, setUserData] = useState("");

  const [isBlock, setBlock] = React.useState(false);

  const [iserror, setIserror] = useState(false);

  const [alertMsg, setAlertMsg] = useState("");

  const [isUpdating, setIsUpdating] = useState(false);

  const [err, setErr] = useState("");

  const [metricList, setMetricList] = useState([]);

  const [metricName, setMetricName] = useState("");

  const [createdAt, setCreatedAt] = useState("");

  const [isSuccess, setIsSuccess] = useState(false);

  const openBlock = () => {

    setBlock(true);

  };

  const closeBlock = () => {

    setBlock(false);

  };

  const [isDelete, setDelete] = React.useState(false);

  const openDelete = () => {

    setDelete(true);

  };

  const closeDelete = () => {

    setDelete(false);

  };

  function createData(sno, metricname, createdat) {

    return {

      sno,

      metricname,

      createdat,

    };

  }

  const rows = [

    createData(1, "Units", "25/03/2021,5:16 PM"),

    createData(2, "Units", "25/03/2021,5:16 PM"),

    createData(3, "Units", "25/03/2021,5:16 PM"),

  ];

  const accessToken = window.localStorage.getItem("creatturAccessToken");

  console.log("token", accessToken);

  useEffect(() => {

    // setIsLoading(true);

    axios

      .post(

        ApiConfig.metricList,

        {},

        {

          headers: {

            token: ` ${accessToken}`,

          },

        }

      )

      .then((response) => {

        // console.log("data",response);

        if (response.data.responseCode !== 200) {

          setIserror(true);

          setIsSuccess(false);

          setAlertMsg(response.data.responseMessage);

        } else {

          setIsSuccess(true);

          setIserror(false);

          // setIsLoading(false);

          setAlertMsg(response.data.responseMessage);

          let result = response.data.result;

          console.log("data", result);

          setMetricList(result.docs);

          console.log("result", result);

          // setIsUpdating(false);

          // let {metricName, createdat} = response.data.result;

          // setUserData({metricName, createdat})

          //SettingsInputAntennaTwoTone

          console.log("result", result);

        }

      })

      .catch((response) => {

        console.log("response", response);

        setIsUpdating(false);

      });

  }, []);

  return (

    <Container maxWidth="xl">

      <Page

        style={{ display: "flex", flexDirection: "column" }}

        title="User Management"

      >

        <Box py={3}>

          <Typography variant="h3" style={{ marginBottom: "8px" }}>

            <strong>METRIC MANAGEMENT</strong>

          </Typography>

          <Divider />

          <Box py={5}>

            <SearchFilter addProps={{ title: "add new" }} />

          </Box>

        </Box>

        <div style={{ overflow: "auto" }}>

          <TableContainer component={Paper}>

            <Table className={classes.table} aria-label="simple table">

              <TableHead>

                <TableRow>

                  <TableCell

                    style={{ color: "white", backgroundColor: "#252d47" }}

                  >

                    S.No.

                  </TableCell>

                  <TableCell

                    align="center"

                    style={{ color: "white", backgroundColor: "#252d47" }}

                  >

                    Metric Name

                  </TableCell>

                  <TableCell

                    style={{ color: "white", backgroundColor: "#252d47" }}

                    align="center"

                  >

                    Created At

                  </TableCell>

                  <TableCell

                    style={{ color: "white", backgroundColor: "#252d47" }}

                    align="center"

                  >

                    Action

                  </TableCell>

                </TableRow>

              </TableHead>

              <TableBody>

                {metricList.map((obj, i) => {

                  console.log(obj);

                  return (

                    <TableRow key={obj.name}>

                      <TableCell component="th" scope="row">

                        {i + 1}

                      </TableCell>

                      <TableCell align="center">{obj.metricName}</TableCell>

                      <TableCell align="center">{obj.createdAt}</TableCell>

                      <TableCell style={{ width: 5 }} align="left">

                    <Box display="flex">

                    <Link

                     to={{

                      pathname: "/viewMatric",

                      state: {

                        data: obj.\_id,

                      },

                    }}

                    component={RouterLink}>

            <Button

              variant="contained"

              color="primary"

              className={classes.button}

            >

              <VisibilityIcon style={{ fontSize: "15px" }} />

            </Button>

          </Link>

                      </Box>

                  </TableCell>

                    </TableRow>

                  );

                })}

              </TableBody>

            </Table>

          </TableContainer>

          <Box display="flex" justifyContent="flex-end">

            <Pagination count={10} shape="rounded" />

          </Box>

        </div>

      </Page>

    </Container>

  );

}

const SearchFilter = ({

  exportProps,

  addProps,

  showTabs,

  searchProps,

  selectedTab,

  tabChange,

  transactionType,

  currency,

  searchBar,

}) => {

  const [selectedDate, setSelectedDate] = React.useState(

    new Date("2014-08-18T21:11:54")

  );

  const handleDateChange = (date) => {

    setSelectedDate(date);

  };

  const [addNew, setAddNew] = useState(false);

  const useStyles = makeStyles({

    table: {

      // minWidth: 700,

    },

    button\_style: {

      width: 100,

    },

  });

  const history = useHistory();

  const [metricName, setMetricName] = useState("");

  const [iserror, setIserror] = useState(false);

  const [alertMsg, setAlertMsg] = useState("");

  const [isUpdating, setIsUpdating] = useState(false);

  const [err, setErr] = useState("");

  const [isSuccess, setIsSuccess] = useState(false);

  const [name, setName] = useState(true);

  const isMountedRef = useIsMountedRef();

  const [isSubmit, setIsSubmit] = useState(false);

  const [formData, setFormData] = useState({

    name: "",

    // lastName: "",

    // stateName: "",

    // cityName: "",

  });

  const \_onInputChange = (e) => {

    const name = e.target.name;

    const value = e.target.value;

    const temp = { ...formData, [name]: value };

    setFormData(temp);

  };

  const auth = useContext(AuthContext);

  const submitHandler = async () => {

    setIsSubmit(true);

    if (metricName !== "") {

      // saveFormData(formData);

      try {

        setIsUpdating(true);

        const accessToken = window.localStorage.getItem("creatturAccessToken");

        const response = await axios.post(

          ApiConfig.metricAdd,

          { metricName: metricName },

          {

            headers: {

              token: ` ${accessToken}`,

            },

          }

        );

        console.log(response);

        if (response.data.responseCode !== 200) {

          setIserror(true);

          setIsSuccess(false);

          setAddNew(false);

          setAlertMsg(response.data.responseMessage);

        } else {

          setIsSuccess(true);

          setIserror(false);

          setAddNew(false);

          auth.userLogIn(true, response.data.result.token);

          // history.push("/otp",email);

          history.push({

            // pathname: '/metric',

            state: { metricName: metricName },

          });

          setAlertMsg(response.data.responseMessage);

        }

        setIsUpdating(false);

      } catch (err) {

        console.log("ERROR", err);

        setIsUpdating(false);

      }

    }

  };

  return (

    <Card raised>

      <CardContent>

        {showTabs && (

          <Box pb={7}>

            <Tabs

              value={selectedTab}

              onChange={tabChange}

              aria-label="simple tabs example"

              indicatorColor="secondary"

            >

              <Tab value="individual" label="Individual" />

              <Tab value="corporate" label="Corporate" />

            </Tabs>

          </Box>

        )}

        <Typography variant="h4" style={{ marginBottom: "8px", marginTop: 10 }}>

          Filter by

        </Typography>

        <Grid container spacing={1} alignItems="center">

          {searchBar && (

            <Grid item md={3}>

              <Typography variant="h3">Search:</Typography>

              <TextField

                style={{ marginTop: "10px" }}

                variant="outlined"

                placeholder="Search "

                fullWidth

                size="small"

              />

            </Grid>

          )}

          {transactionType && (

            <Grid item md={3}>

              <Typography variant="h3">Transaction type</Typography>

              <TextField

                variant="outlined"

                size="small"

                placeholder="transaction Type"

              />

            </Grid>

          )}

          {currency && (

            <Grid item md={3}>

              <Typography variant="h3">Currency</Typography>

              <TextField

                variant="outlined"

                size="small"

                placeholder="currency"

              />

            </Grid>

          )}

          <Grid item md={2} lg={2} sm={12} xs={12}>

            <Typography variant="h3">From Date</Typography>

            <MuiPickersUtilsProvider utils={DateFnsUtils}>

              <form noValidate>

                <TextField

                  variant="outlined"

                  size="small"

                  fullWidth

                  id="date"

                  // inputLabelProps={{ style: { fontSize: 10 } }}

                  inputProps={{ style: { fontSize: 14 } }}

                  type="date"

                  defaultValue="2017-05-24"

                  style={{ paddingTop: 10 }}

                  InputLabelProps={{

                    shrink: true,

                  }}

                />

              </form>

            </MuiPickersUtilsProvider>

          </Grid>

          <Grid item md={2} lg={2} sm={12} xs={12}>

            {" "}

            <Typography variant="h3">To Date</Typography>

            <MuiPickersUtilsProvider utils={DateFnsUtils}>

              <form noValidate>

                <TextField

                  variant="outlined"

                  size="small"

                  fullWidth

                  id="date"

                  // inputLabelProps={{ style: { fontSize: 10 } }}

                  inputProps={{ style: { fontSize: 14 } }}

                  type="date"

                  defaultValue="2017-05-24"

                  style={{ paddingTop: 10 }}

                  InputLabelProps={{

                    shrink: true,

                  }}

                />

              </form>

            </MuiPickersUtilsProvider>

          </Grid>

          <Grid item md={4} lg={4} sm={12} xs={12}>

            <Box>

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

              <Box style={{ marginTop: 10 }}>

                <FormControl size="small" variant="outlined" fullWidth>

                  <OutlinedInput

                    placeholder="search by name"

                    inputProps={{ style: { fontSize: 14 } }}

                    endAdornment={

                      <InputAdornment position="end">

                        <SearchIcon

                          aria-label="toggle password visibility"

                          edge="end"

                        ></SearchIcon>

                      </InputAdornment>

                    }

                  />

                </FormControl>

              </Box>

            </Box>

          </Grid>

          {searchProps && (

            <Grid item md={4}>

              <Box mt={4}>

                <TextField

                  variant="outlined"

                  placeholder={searchProps}

                  size="small"

                />

              </Box>

            </Grid>

          )}

          <Grid item md={2} lg={2} sm={12} xs={12}>

            <Box

              style={{ marginTop: 35 }}

              display="flex"

              alignItems="flex-end"

              justifyContent="space-around"

            >

              <Button variant="contained" color="primary">

                Search

              </Button>

              <Button

                variant="contained"

                color="primary"

                style={{ marginLeft: "3px", marginRight: "3px" }}

              >

                Reset

              </Button>

              {exportProps && (

                <Button

                  variant="contained"

                  color="primary"

                  onClick={exportProps.onClick}

                >

                  {exportProps.title}

                </Button>

              )}

            </Box>

          </Grid>

        </Grid>

        <Box display="flex" justifyContent="flex-end" py={4}>

          {addProps && (

            <Link href={addProps.link}>

              <Button

                onClick={() => setAddNew(true)}

                variant="contained"

                color="primary"

              >

                {addProps.title}

              </Button>

              <Modal

                isOpen={addNew}

                onClose={false}

                onRequestClose={() => setAddNew(false)}

                //  className={classes.paper}

                style={{

                  overlay: {

                    position: "fixed",

                    width: "100%",

                    top: 0,

                    // width:0,

                    left: 0,

                    right: 0,

                    bottom: 0,

                    backgroundColor: "transparent",

                  },

                  content: {

                    position: "absolute",

                    top: 150,

                    left: "40%",

                    // right: 300,

                    width: "40%",

                    border: "1px solid #ccc",

                    height: 250,

                    overflow: "auto",

                    WebkitOverflowScrolling: "touch",

                    borderRadius: 5,

                    outline: "none",

                    padding: "20px",

                    opacity: "none",

                  },

                }}

              >

                <Box>

                  <Box justifyContent="center" display="flex">

                    <Typography variant="h2">Create New Metric</Typography>

                  </Box>

                  <Divider />

                  <Box justifyContent="center" display="flex" pt={3}>

                    <Typography variant="h3">Enter Metric Name</Typography>

                  </Box>

                  <Box mb={2} justifyContent="center" display="flex">

                    <TextField

                      style={{ marginTop: 5, width: 250 }}

                      variant="outlined"

                      placeholder="Enter Metric Name"

                      size="small"

                      fullWidth

                      name="metricName"

                      size="small"

                      onChange={(e) => setMetricName(e.target.value)}

                      error={metricName !== "" && !isValidMetric(metricName)}

                      helperText={

                        metricName !== "" &&

                        !isValidMetric(metricName) &&

                        "Please enter valid metric name"

                      }

                    ></TextField>

                  </Box>

                  <Box justifyContent="center" display="flex">

                    <Button

                      variant="contained"

                      color="primary"

                      // className={classes.button\_style}

                      // onClick={() => setModalIsOpen(false)}

                      onClick={submitHandler}

                    >

                      Submit

                    </Button>

                    <Button

                      variant="contained"

                      color="primary"

                      // className={classes.button\_style}

                      onClick={() => setAddNew(false)}

                      style={{ marginLeft: 10 }}

                    >

                      Cancel

                    </Button>

                  </Box>

                </Box>

              </Modal>

            </Link>

          )}

        </Box>

      </CardContent>

    </Card>

  );

};

import React,{useEffect,useState} from "react";

import {

  Container,

  Divider,

  Box,

  Card,

  Grid,

  CardContent,

  Typography,

  Button,

  Link,

} from "@material-ui/core";

import Page from "src/component/Page";

import axios from "axios";

import ApiConfig from "src/config/APIConfig";

const Row = ({ field, value, image }) => (

  <Grid container md={12}>

    <Grid item md={5}>

      <Box display="flex" justifyContent="space-between" pr={4}>

        <Typography variant="h3">

          {" "}

          <strong>{field}</strong>

        </Typography>

      </Box>

    </Grid>

    <Grid item md={7}>

      <Typography variant="body1">{value}</Typography>

      {image && <img src={image} alt="comp" width='90px'/>}

    </Grid>

  </Grid>

);

const ViewMatric = (props) => {

  const matricId = props.location && props.location.state.data;

  const accessToken = window.localStorage.getItem("creatturAccessToken");

  const [reviewDetails, setReviewDetails] = useState({

    Name :"",

    Date:"",

  });

  useEffect(() => {

    let payload={

        metricId : matricId,

    }

    // setIsLoading(true);

    // console.log(productId, accessToken);

    axios

      .post(ApiConfig.particularMetric,payload, {

        headers: {

          token: accessToken,

        },

      })

      .then((response) => {

        if (response.data.responseCode !== 200) {

        } else {

          console.log(response.data.result);

          setReviewDetails({

            ...reviewDetails,

                        Date:response.data.result.createdAt,

            Name : response.data.result.metricName,

          });

        }

      })

      .catch((response) => {

        // setIsUpdating(false);

        console.log('response', response);

      });

  }, []);

  return (

    <>

      <Container maxWidth="xl">

        <Page

          style={{ display: "flex", flexDirection: "column" }}

          title="View Earning"

        >

          <Box pt={3} mb={8}>

            <Typography variant="h3" style={{ marginBottom: "10px" }}>

              <strong>VIEW MetricManagement</strong>

            </Typography>

            <Divider />

          </Box>

          <Card>

            <CardContent>

              <Grid container md={6} direction="column" spacing={3} style={{paddingBottom:'50px'}}>

                <Grid item>

                  {" "}

                  <Row field="Date :" value={reviewDetails.Date} />

                </Grid>

                <Grid item>

                  {" "}

                  <Row field="Name :" value={reviewDetails.Name} />

                </Grid>

                {/\* <Grid item>

                  {" "}

                  <Row field="Currency Type :" value={reviewDetails.CurrencyType} />

                </Grid>

                <Grid item>

                  {" "}

                  <Row field="Actual Amount :" value={reviewDetails.Amount} />

                </Grid>

                <Grid item>

                  <Row field="System Fee :" value="NA" />

                </Grid>

                <Grid item>

                  <Row field="Amount :" value="150 USD" />

                </Grid>

                <Grid item>

                  <Row field="Status :" value={reviewDetails.status} />

                </Grid> \*/}

                <Grid item>

                    <Link href='/metric'>

                    <Button variant='contained' color='primary' size='large'>Back</Button>

                    </Link>

                </Grid>

              </Grid>

            </CardContent>

          </Card>

        </Page>

      </Container>

    </>

  );

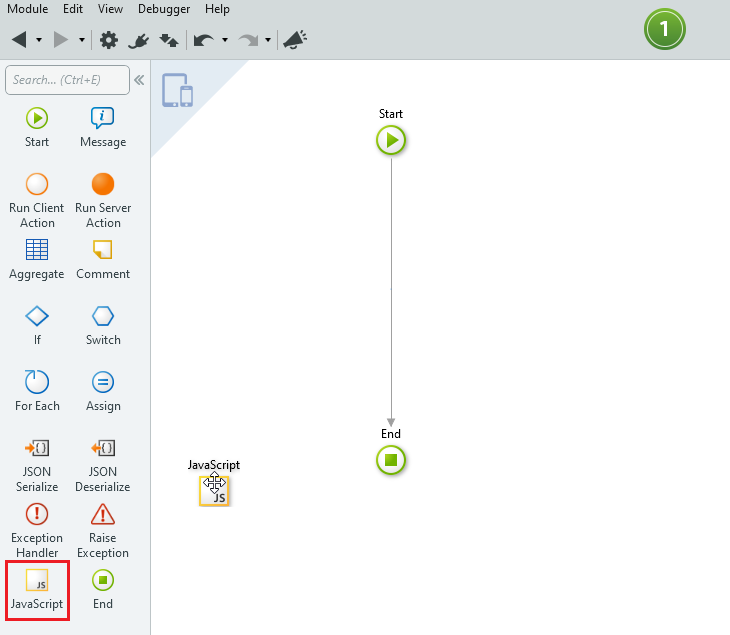
};

export default ViewMatric;

**CHAPTER 5**

**Extend Your Mobile and Reactive Apps Using JavaScript**

JavaScript code can be used in client actions of Reactive and Mobile apps through the **JavaScript element**, which you can drag from the Toolbox to the action flow.



The JavaScript element allows extending the OutSystems capabilities by using JavaScript code. In the code editor you can type regular JavaScript code, such as:

* Declare variables and assign values to them
* Invoke user-defined or built-in functions
* Call client actions, either synchronously or asynchronously, etc.

The code editor window, opened by double-clicking the element or the element's "JavaScript" property, has auto-complete support for JavaScript keywords and predefined JavaScript objects, as well as for OutSystems available client actions and roles. It also has syntax highlighting and checks the JavaScript code for errors and warnings. The TrueChange pane displays eventual JavaScript syntax errors:

**5.1 Website Setup Menu**

The Website Setup menu is where you will spend the most time as a developer. These menus provide access to pages that let you create and configure components and services.

Once you create a custom object, you can edit the definition of the object via the Custom Object page.

The Custom Object page provides links for adding custom fields, validations to enforce data integrity rules, database triggers, and custom buttons or links to the object's page layouts. You can also modify the attributes of standard fields, buttons, links or layouts for both the page and search dialogs, as well as add new page layouts or assign record types.

As an example, when you add a new field to a custom object, a wizard walks you through a number of steps, including:

* Selecting a field type.
* Giving the field a label, name, help text, a default value, and potentially other attributes, such as the length of the field or whether a value is required.
* Assigning security settings to the field.
* Adding the field to existing page layouts.

Depending on the type of field in focus, the wizard may include other pages for other relevant metadata attributes.

**CHAPTER 6**

**TESTING**

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

An early start to testing reduces the cost and time to rework and produce error-free software that is delivered to the client. However in Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the deployment of the software.

It also depends on the development model that is being used. For example, in the Waterfall model, formal testing is conducted in the testing phase; but in the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.

Testing is done in different forms at every phase of SDLC −

* During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
* Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
* Testing performed by a developer on completion of the code is also categorized as testing.
* Testing must include UI testing, functional testing, regression testing, integration testing, system testing and system integration testing.
* Automation testing can also be enforced on using tools like HP Unified Functional Testing (UFT ) and Selenium.
* A tester needs to be cautious during UI testing as most of the web pages on the Salesforce platform are Visual Force pages. The dynamic nature of visual force pages need to be paid special attention as all the elements of a webpage may not be loaded at one go.
* Testers need to create functional flows including positive and negative flows to cover the entire functionality of an application.
* Workflows using various user roles must be constructed and tested.
* Test cases need to be documented using a test management tool like HP ALM.

Test Data needs to be prepared for validating the reports functionality.

#### 6.1 Exploratory Testing

**Exploratory Testing in Salesforce would involve the following best practices:**

* Testing should involve validating the consistency of data across multiple screens.
* UI Testing must involve documented test cases as per the requirement document.
* Testing should involve negative test flows, such as deleting the default data generated and validating the behaviour of an application.
* Testing should involve user input validation on the form fields.
* [Cross browser compatibility testing](https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/) needs to be performed to ensure if the rendering of data is correct across multiple browsers.
* Testing must include Maximum length validation for each of the editable input fields along with the invalid data validation.
* Testing must also include error message validation when invalid data is passed onto the applications.
* Amount field validation on banking applications using Boundary Value Analysis technique needs to be performed with proper diligence.
* Reports and dashboard testing need to be paid special attention to various test data parameters.
* Testing should include the entire application flow, along with individual functional flows.
* Multiple permutations and combinations of functional flows can be tested for positive and [negative testing](https://www.softwaretestinghelp.com/what-is-negative-testing/).
* API testing for integrated third-party applications needs to be performed.
* Identify the default Salesforce functionalities that come in the way of customized features and coordinate with the developers.

#### 6.2 Test Automation

Automated functional testing of SalesForce is a challenging one as most of the web pages are dynamic in nature on the SalesForce platform. Hence, SalesForce demands automation testers to build robust automation framework to sustain in the future. Also, there can be frequent updates to the applications as they are on cloud applications.

**Test Automation on Salesforce can be achieved using any of the following tools:**

* Selenium web driver
* HP Unified Functional Testing (UFT)
* Test Frameworks, such as Cucumber
* Provar

#### 6.3 Load Testing

Load testing involves testing the behavior of an application under varying loads. SalesForce.com is a highly scalable platform built for handling a large number of users. Salesforce.com is tested by the platform developers themselves for performance bottlenecks.

However, load testing becomes essential when a newly introduced piece of code yields performance bottlenecks that have to be addressed. Load Testing on Salesforce platform can be performed using performance testing tools such as HP LoadRunner and Apache JMeter.

#### 6.4 Security Testing

Security testing on the Salesforce platform is usually done by SalesForce development team. Before placing a request for a security test, it is best to review the ‘Application and Network Vulnerability Assessment Summaries’ provided by Salesforce.

After reviewing the summary, if a security test is still required, then a Security Assessment Test can be scheduled with the Salesforce team.

**CHAPTER 7**

**REFERENCES**

The following references have been used by me, during all the phases of the project:

1. Shailesh Kumar Shivakumar, “Complete Guide to Digital Project Management”
2. Ralph L. Kliem, PMP (2013) “Creative, Efficient, and Effective Project Management”
3. Stefano Tonchia Foreword by Russell D. Archibald, “Industrial Project Management”

Department DPIA, University of Udine, Udine, Italy

1. Ofer Zwikael and John R. Smyrk, “Project Management ”
2. Chitram Lchman(2011), A Practical Approach to Industrial and Commercial Project Management.
3. Abraham, J. (1996), “Computers in modernising Library  Information System and  Services: Perspectives of Library Automation”, International Library Movement” vol.  18.
4. Breeding, Marshall(2009), “LibraryAutomation in a Difficult Economy”, Computers  in Libraries,
5. Lynne Porat (2001) “Automation of interlibrary loan services: effects on the patron  and the library”, Interlending & Document Supply,
6. ["Salesforce.com Launches The Service Cloud, A Customer Service SaaS Application | TechCrunch"](http://www.techcrunchit.com/2009/01/14/salesforcecom-launches-the-service-cloud-a-customer-service-saas-application/)*. Techcrunchit.com. January 14, 2009*. Retrieved February 1, 2012*.*
7. Claude H. Maley (2012) NY “Project Management Concepts, Methods, and Technique ”
8. Mahmood, Khalid and Khan,  Muhammad Ajmal (2008),  “Volunteer Endeavors to  Promote ICT in a Developing Country: the case of  Library Automation  Group”, Information Development, Vol. 24 no. 2, p135­142.
9. Hane,  Paula  J (2008),  “Advances  in  Digital  Reading,  Enterprise  Search,  and  Library  Automation”, Information Today, Vol. 25, No.10, p7­13
10. Kimber, Richard T (1996), “Automation in Libraries”, Oxford Pergamon.
11. Groenewegen, Hans (2004), “Four decades of library automation recollections and  reflections”, AustralianLibraryJournal, Vol. 53, no.1 1, p39­53.
12. Haravu, L.J (1993), “Library automation and networking in lndia: An overview of  recent developments”, Annals of Library Science & Documentation, vol.40, no.1, p  32­40.
13. Hans Ottosson (2013) “Practical Project Management for Building and Construction”
14. Kasiviswanadhan, S (1998), “A practical approach to library automation”, Automation  in Libraries and Information Retrieval Units: RlLISAR Bulietin, vol.4, no.3
15. Adedeji B. Badiru (2019) imprint CRC Press Project Management Systems, Principle, and Applications, Second Edition.