IVENDOR(P2P)

A PROJECT REPORT
Submitted By

Ayush Tyagi
University Roll No 1900290149030

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Ms. Neelam Rawat
KIET Group of Institution, Ghaziabad

to the

DEPARTMENT OF COMPUTER APPLICATIONS

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW
(Formerly Uttar Pradesh Technical University, Lucknow)

(july-21)

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: Ayush Tyagi

Roll. No.: 19002901490370

Branch: Master of Computer Applications

(Candidate Signature)

CERTIFICATE

Certified that **Ayush Tyagi**(**1900290149030**) has carried out the project work presented in this report entitled "**Ivendor(P2P)**" 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.

Ms.Neelam Verma **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:

ABSTRACT

Fieldwork management can be a complicated and complex process nowadays, especially when it involves sophisticated techniques with names like probabilistic sampling, Box-Jenkins, or conjoint analysis. There is a need for a guide that will help managers determine the dimensions of the task, the resources that must be summoned, the data required, and other key elements. Here, from the head of market research at General Motors, is such a guide in outline form. Vincent P. Barabba walks the reader through the five major stages of the research effort. Each stage is represented in one table, and together the tables combine into a mammoth foldout for easy perusal. The stages are: 1. Assess the market information needs. 2. Measure the marketplace. 3. Store, retrieve, and display the data. 4. Describe and analyze market information. 5. Evaluate the research and assess its usefulness. Comprehensiveness is a hallmark of the "encyclopedia." Barabba lists the limitations of this kind of research: "Very expensive; responses in this artificial environment may not reflect responses in the market; complicated logistics; development of test product can be time consuming and expensive."

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. Ashish kumar** 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 **Ms. Neelam Rawat** 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 **"Ivendor(P2P)"**. 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.

Ayush tyagi
1900290149030

TABLE OF CONTENTS

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.

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.

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 cap crossed 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 (#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, you can easily enter the market and start investing.

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

Techolnogy 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.

Create the following Custom Objects

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

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)

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

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.

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.

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.

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.

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

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.

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

Login Screen

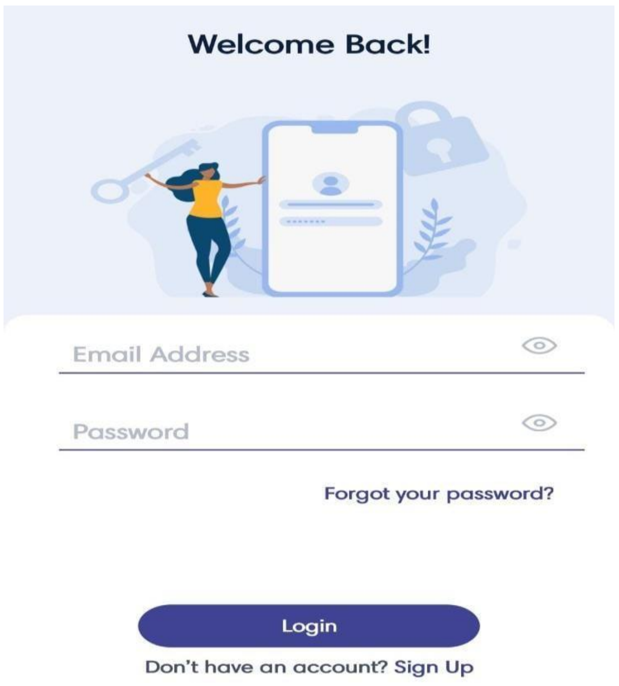
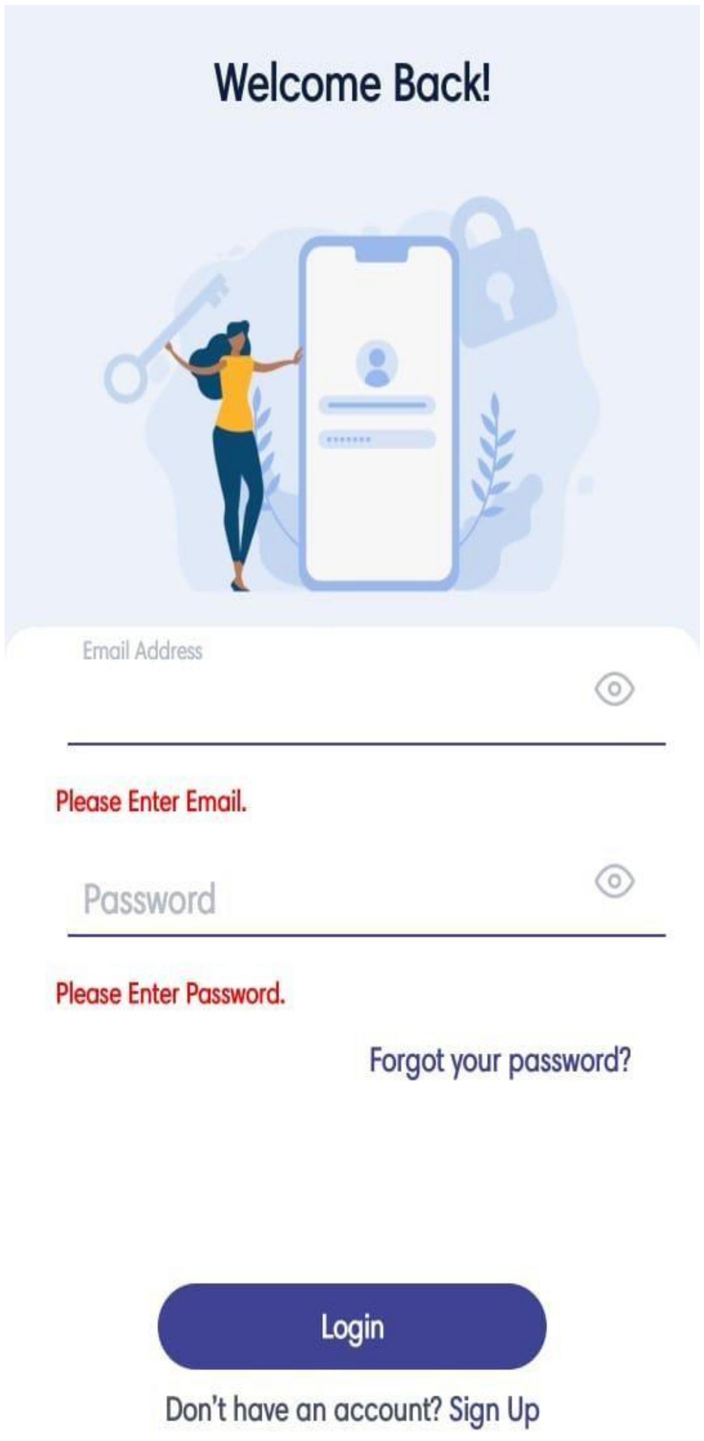


Figure 1


Welcome Back!

Email Address

Please Enter Email.

Password

Please Enter Password.

Forgot your password?

Don't have an account? Sign Up

Figure 2

Dashboard Screen

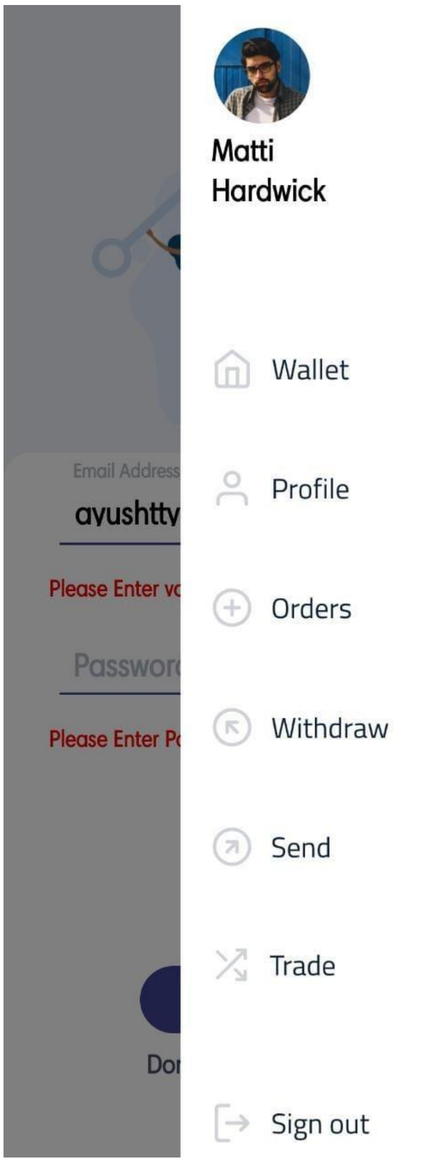


Figure 3

Bit Coin Screen

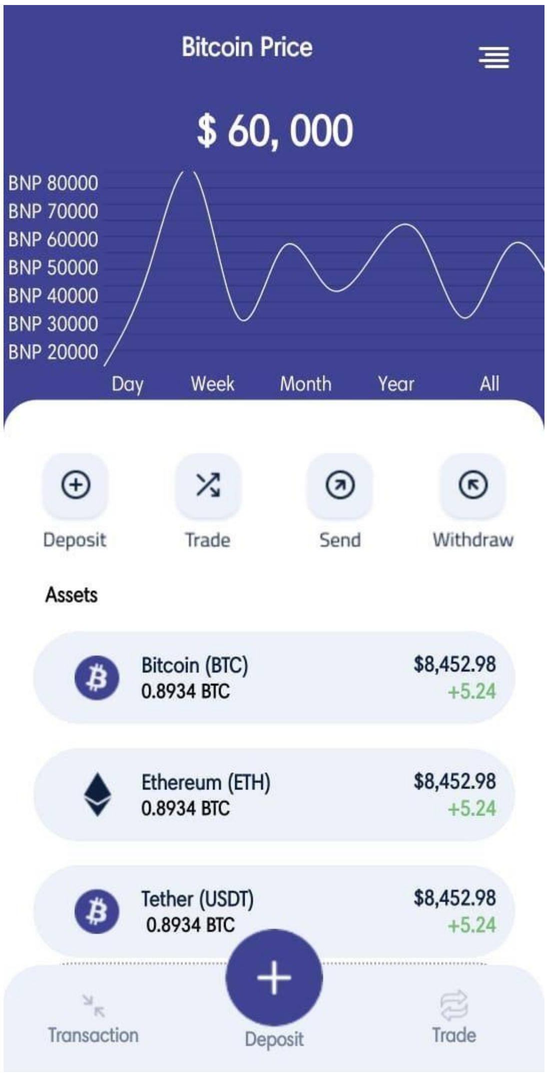
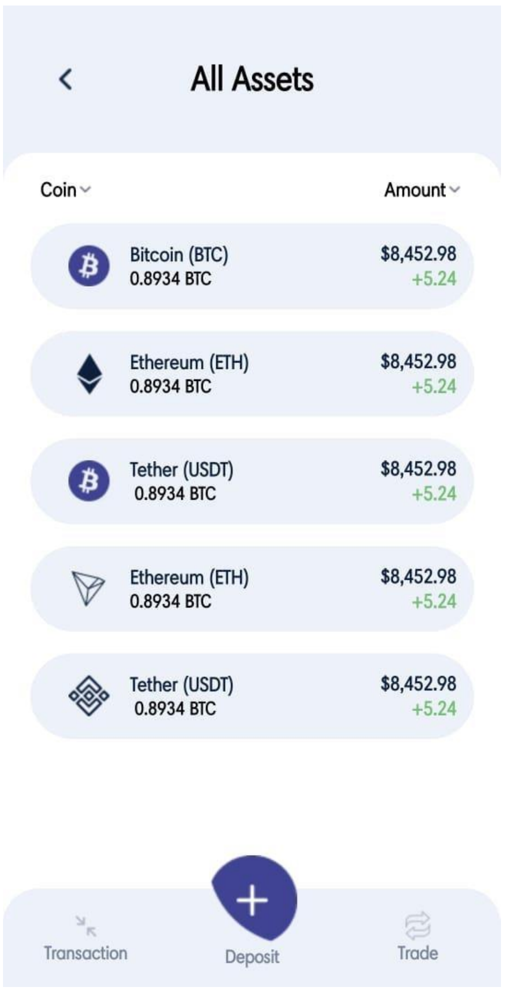


Figure 4

,

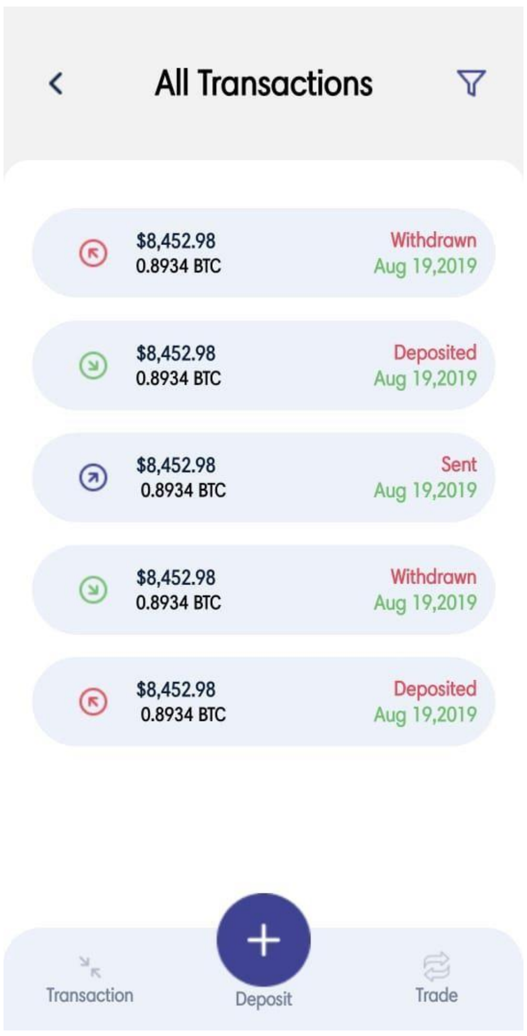
All Assets


All Assets

Coin	Amount
Bitcoin (BTC) 0.8934 BTC	$8,452.98 +5.24
Ethereum (ETH) 0.8934 BTC	$8,452.98 +5.24
Tether (USDT) 0.8934 BTC	$8,452.98 +5.24
Ethereum (ETH) 0.8934 BTC	$8,452.98 +5.24
Tether (USDT) 0.8934 BTC	$8,452.98 +5.24

Figure 5

Al l Transactions

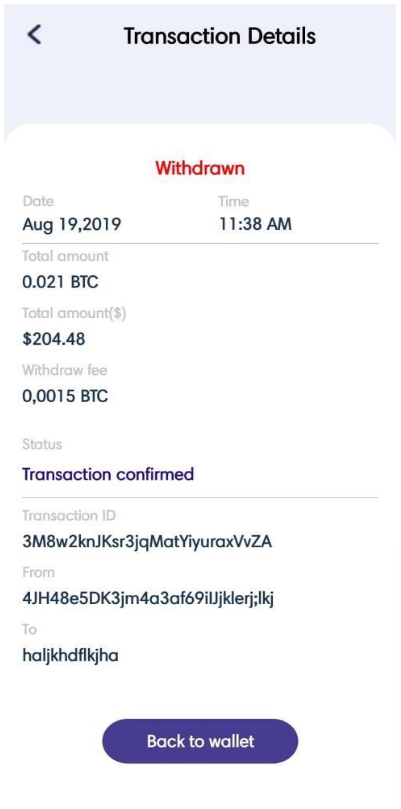


All Transactions

$8,452.98	0.8934 BTC	Withdrawn	Aug 19,2019
$8,452.98	0.8934 BTC	Deposited	Aug 19,2019
$8,452.98	0.8934 BTC	Sent	Aug 19,2019
$8,452.98	0.8934 BTC	Withdrawn	Aug 19,2019
$8,452.98	0.8934 BTC	Deposited	Aug 19,2019

Figure 6

Transaction Details


Transaction Details

Withdrawn

Date
Aug 19,2019

Time
11:38 AM

Total amount
0.021 BTC

Total amount($)
$204.48

Withdraw fee
0,0015 BTC

Status
Transaction confirmed

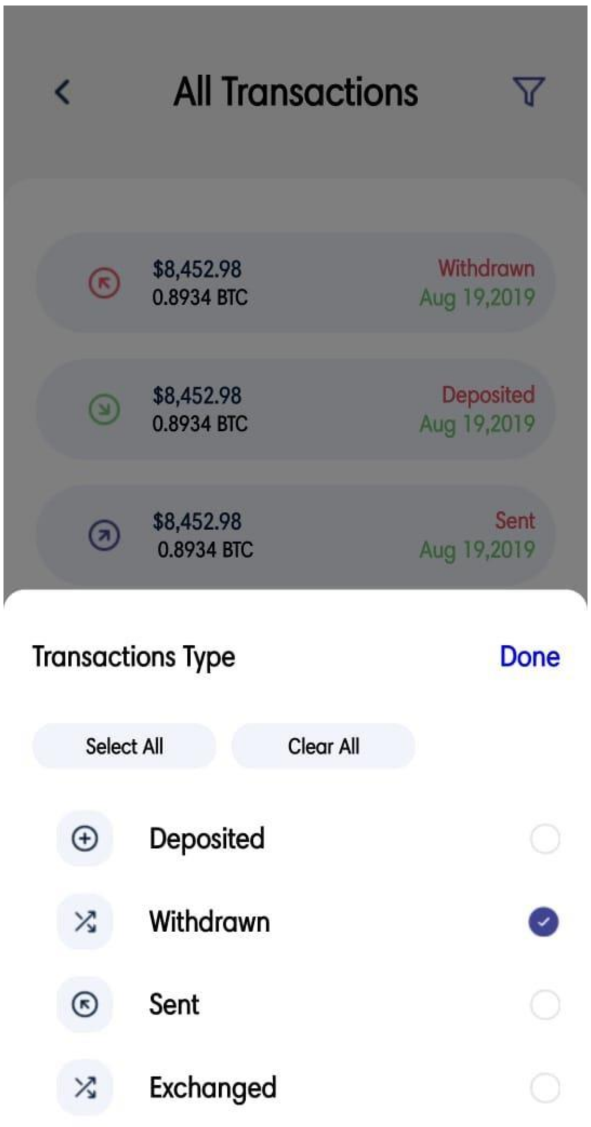
Transaction ID
3M8w2knJKsr3jqMatYiyuraxVvZA

From
4JH48e5DK3jm4a3af69iIJjklerj;lkj

To
haljkhdflkjha

Figure 7

Trancation Type


All Transactions

$8,452.98	0.8934 BTC	Withdrawn Aug 19,2019
$8,452.98	0.8934 BTC	Deposited Aug 19,2019
$8,452.98	0.8934 BTC	Sent Aug 19,2019

Transactions Type

Deposited

[✓ Withdrawn]

Sent

Exchanged

Figure 8

Review Withdrawl

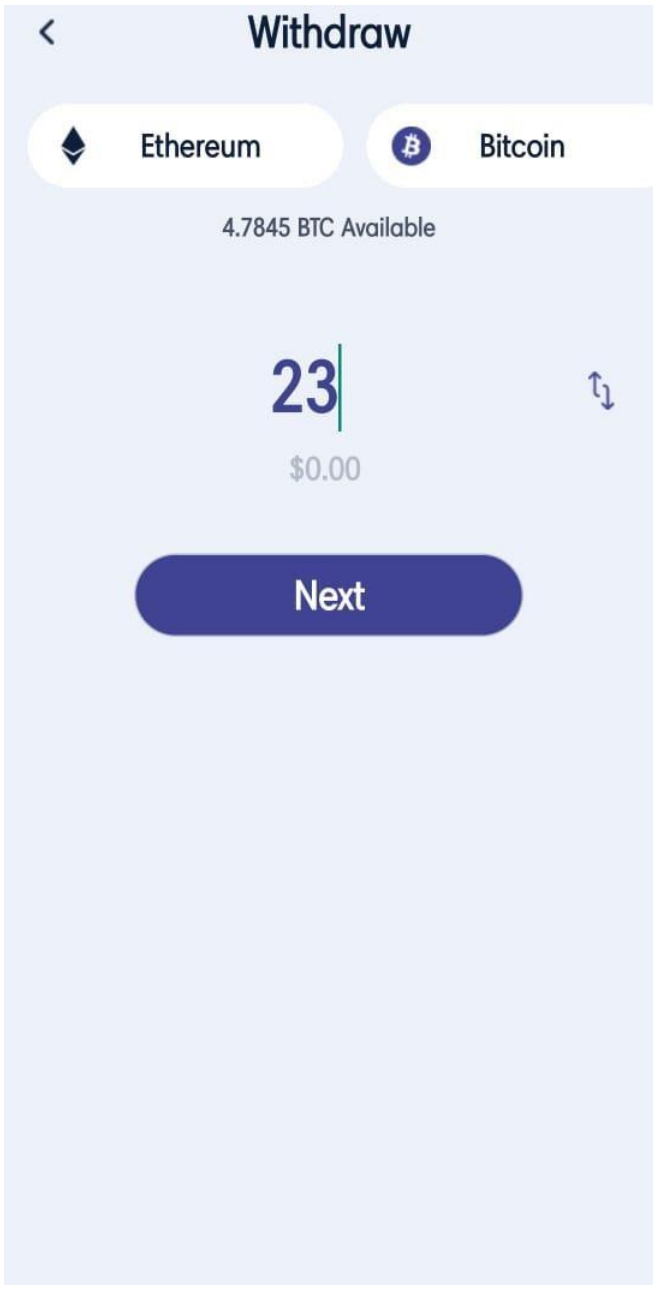


Figure 9

WithdrawII

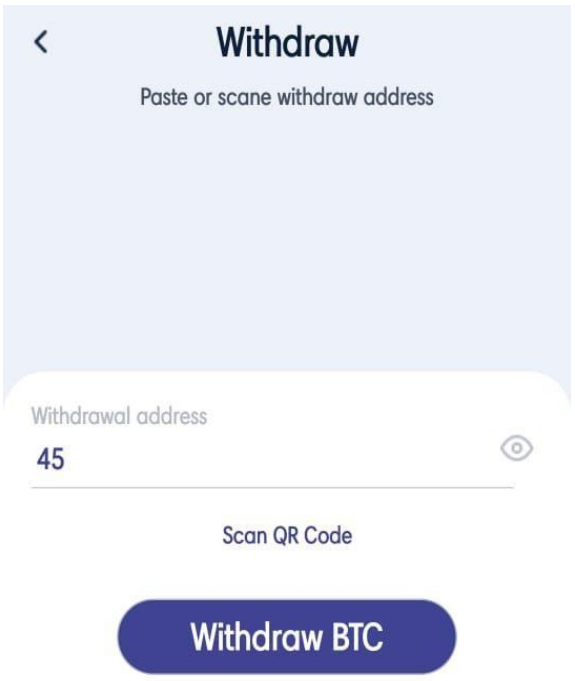


Figure 10

Scan QR Code

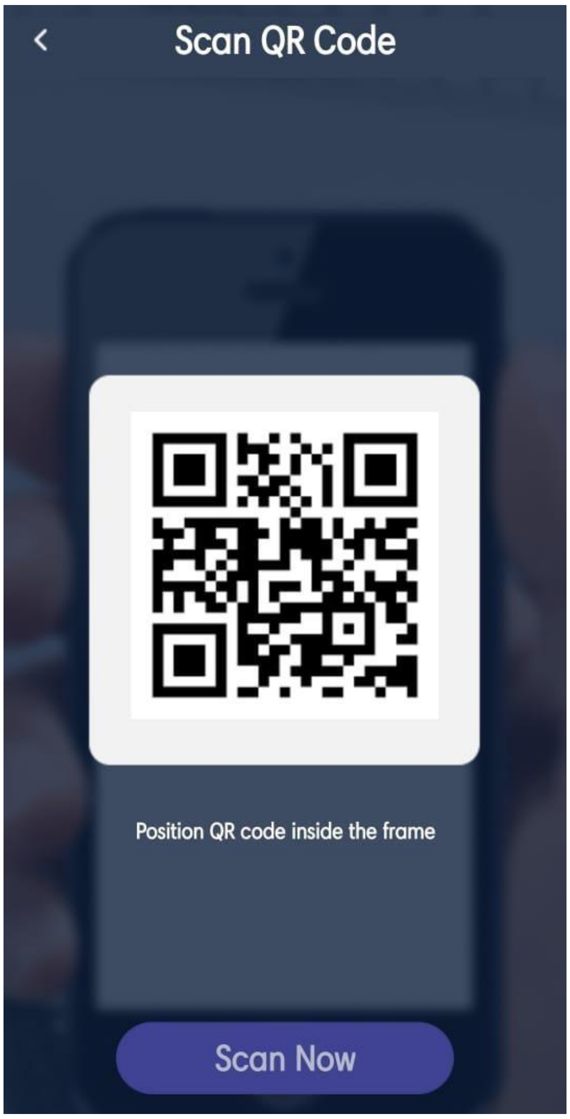


Figure 11

Deposite Coin

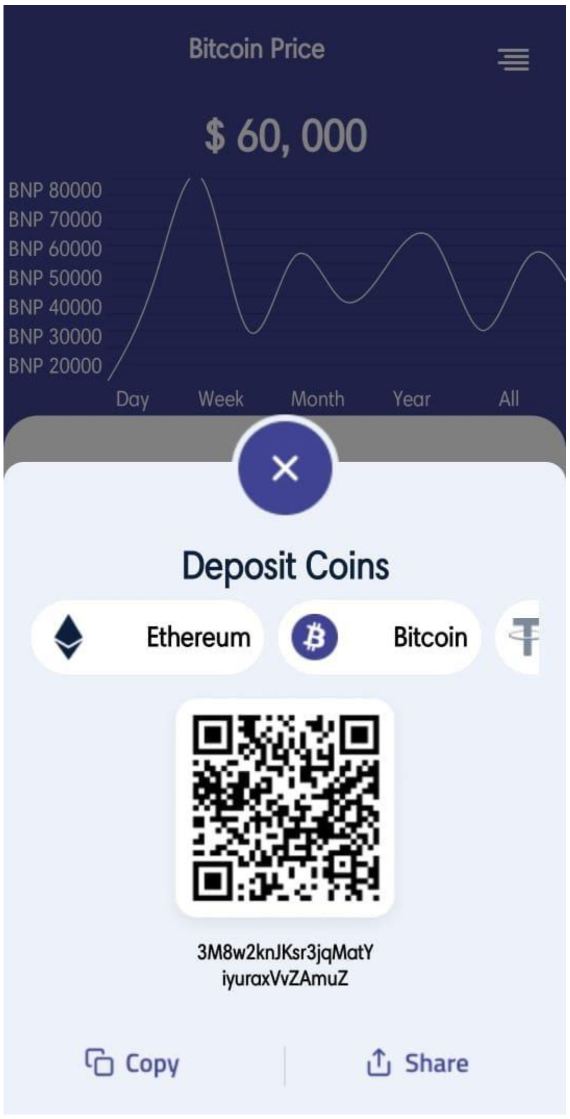
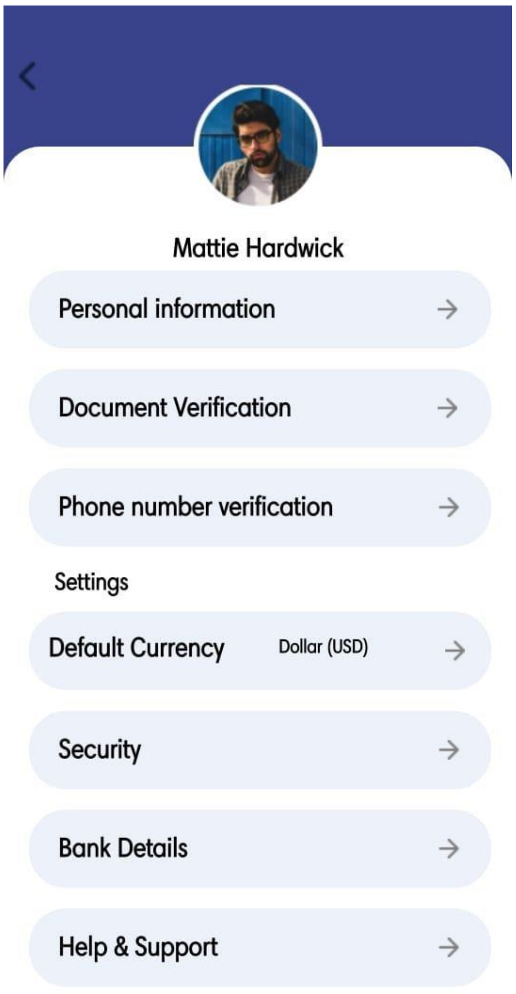


Figure 12

Profile


Mattie Hardwick

Personal information

Document Verification

Phone number verification

Settings

Default Currency Dollar (USD)

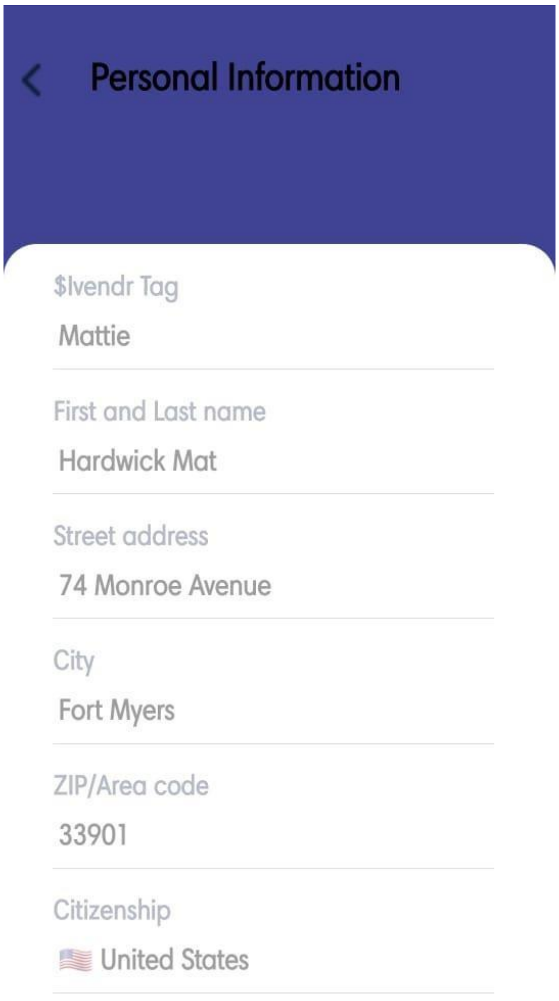
Security

Bank Details

Help & Support

Figure 13

Personal Information


Personal Information

$Ivendr Tag
Mattie

First and Last name
Hardwick Mat

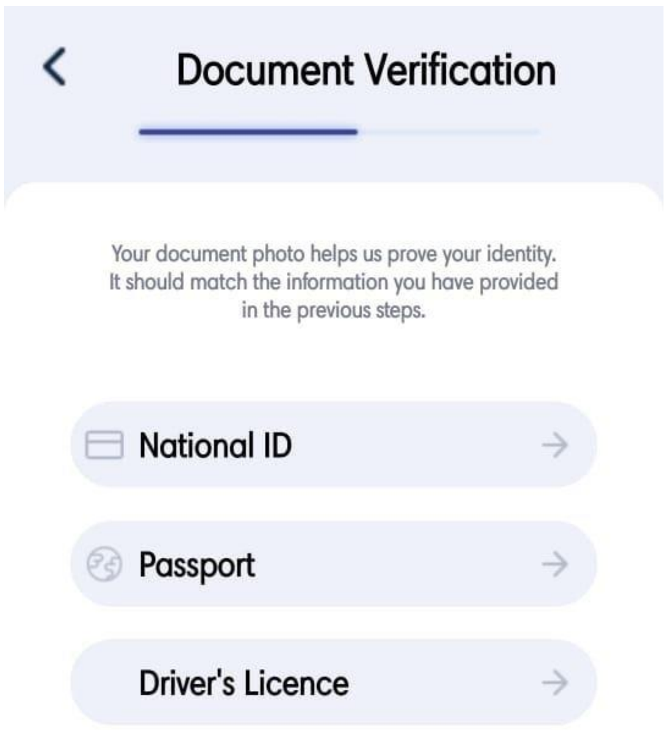
Street address
74 Monroe Avenue

City
Fort Myers

ZIP/Area code
33901

Citizenship
🇺🇸 United States

4.14 Document Verification



Document Verification

Your document photo helps us prove your identity. It should match the information you have provided in the previous steps.

Figure 15

4.13 Verify Number

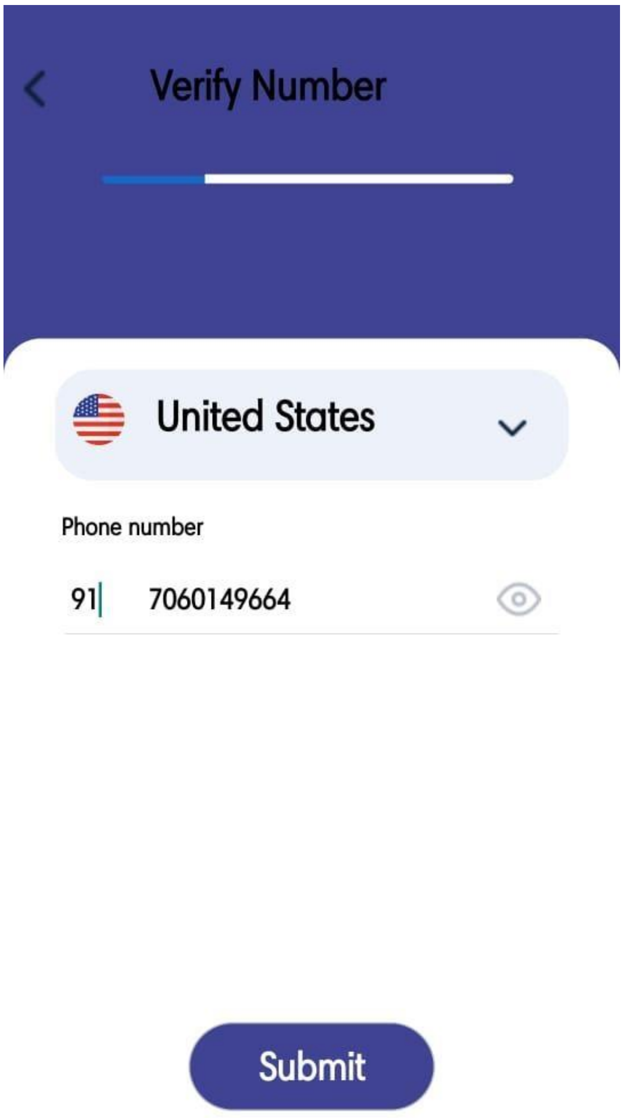


Figure 16

Login Screen

```
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="40%">
          <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 display="flex" alignItems="flex-start"> */}
  <TextField
    placeholder="Enter Mobile Number"
    variant="outlined"
    // 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)}
    helperText={
      phoneNumber !== "" &&
      !isValidContact(phoneNumber) &&
      "Please enter valid mobile number"
    }

  ></TextField>
  {/* {number ? (
  <span
    style={{
      variant: "subtitle2",
      color: "red",
      marginBottom: "10px",
    }}
  >
    Please enter valid phone number
  </span>
) : (
  " "
)} */}
  {/* </Box> */}
</Grid>
<Grid item lg={12} md={12} sm={12} xs={12}>
  <FormControl
    style={{ width: "100%", marginTop: 15 }}
    size="small"
    variant="outlined"
  >
    <OutlinedInput
      placeholder="Enter Password"

      id="outlined-adornment-password"
      type={values.showPassword ? "text" : "password"}
```

```
         value={values.password}
         // 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)}
         helperText={
          password !== "" &&
          !isValidPassword(password) &&
          "At least one uppercase letter, one lowercase letter, one number and one special char
acter"
         }
        />
        {password !== "" &&
         !isValidPassword(password) && (
          <span
           style={{
            variant: "subtitle2",
            color: "red",
            marginBottom: "10px",
           }}
          >
           Minimum eight characters
           one lowercase letter, one number
          </span>
         )}
       </FormControl>
      </Grid>
```

```
<Grid
  container
  spacing={0}
  style={{ marginTop: 5, display: "flex", alignItems: "center" }}
>
  <Grid item lg={6} md={6} sm={12} xs={12}>
    <Box
      display="flex"
      justifyContent="flex-start"
      alignItems="center"
    >
      <FormControlLabel
        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>Forget Password?</Typography>
      </Link>
    </Box>
  </Grid>
</Grid>
<Grid item lg={12} md={12} sm={12} xs={12}>
  <Box mt={6}>
    <Button
      variant="contained"
      style={{
        width: "100%",
        backgroundColor: "#252d47",
        color: "white",
      }}
      onClick={() => history.push("/dashboard")}
    // onClick={submitHandler}
    >
```

```
        </Button>
        <Typography >{err}</Typography>
       </Box>
      </Grid>
     </Grid>
    </Box>
   </Box>
  </Box>
 );
}

export default Login;
```

DashBoard Screen

```
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 axios from "axios";
import ApiConfig from "src/config/APIConfig";
export default function (props) {
const[totalDeposit,setTotalDeposit]=useState();
 const[totalWithdrawAmount,setTotaltWithdrawAmount]=useState();
 const accessToken = window.localStorage.getItem("creatturAccessToken");
const getTotalDeposit=()=>{
 console.log(ApiConfig.getTotalDeposit)
 axios.get(ApiConfig.getTotalDeposit, {

  headers:{
   token:accessToken,
  }
 })
 .then((response) => {
   if (response.data.responseCode !== 200) {

   } else {
    setTotalDeposit(response.data.result.totalDepositBalance)
     // setDisplayname
     // console.log(result)
     console.log(response)

     // else setHistory(depositFilter);

   }
   // setIsLoading(false);
 })
 .catch((response) => {
   // setIsUpdating(false);

   console.log("response", response);
 });
}
const getTotalWithdrawalAmount=()=>{
 axios.get(ApiConfig.getTotalWithdrawAmount, {

  headers:{
    token:accessToken,
  }

 })
 .then((response) => {
   if (response.data.responseCode !== 200) {

   } else {
    setTotaltWithdrawAmount(response.data.result.totalWithdrawlBalance)
     // setDisplayname
     // console.log(result)
     console.log(response)

     // else setHistory(depositFilter);

   }
   // setIsLoading(false);
 })
 .catch((response) => {
   // setIsUpdating(false);
   console.log("response", response);
  });
}
 useEffect(() => {
  // setIsLoading(true);
  getTotalDeposit();
  getTotalWithdrawalAmount();
 }, []);
 const cardData1 = [
  {
    title: 'TOTAL KYC',
    amount: 56,
    icon: <FileCopyIcon fontSize='large'/>
  },
  {
    title: 'TOTAL INVESTMENT PRODUCT',
    amount: '08',
    icon: <BarChartIcon fontSize='large'/>
  },
  {
    title: 'TOTAL WITHDRAWAL REQUEST',
    amount: '100',
    // icon:<AttachMoneyIcon fontSize='large'/>
    icon: <BarChartIcon fontSize='large'/>

  }
 ];
 const cardData2 = [
  {
    title: 'TOTAL TRANSACTION',
    amount: 65,
    icon:<LocalAtmIcon fontSize='large'/>
  },
  {
    title: 'TOTAL DEPOSIT BALANCE',
    amount: totalDeposit,
    icon:<MoneyIcon fontSize='large'/>
  },
  {
    title: 'TOTAL REGISTER USER',
    amount: 85,
    icon:<HowToRegIcon fontSize='large'/>
  }
 ];
```

```
const cardData3 = [
  {
    title: 'TOTAL WITHDRAWAL AMOUNT',
    amount: totalWithdrawAmount,
    icon:<AttachMoneyIcon fontSize='large'/>
  },
  {
    title: 'PENDING WITHDRAWAL REVIEW ',
    amount: 38,
    icon: <AccountBalanceIcon fontSize='large' />
  },
  {
    title: 'PENDING DEPOSIT REVIEW',
    amount: 18,
    icon: <TransformIcon fontSize='large' />
  }
];
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>
 );
}
```

Bit Coin Screen

```
import {
  Typography,
  Box,
  Paper,
  Button,
  Divider,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  TableBody,
  Table,
  TextField,
  makeStyles,
  Grid,
  Link,
  Card,
  CardContent,
  Container,
  FormControl,
  OutlinedInput,
  InputAdornment
} from "@material-ui/core";
import SearchIcon from "@material-ui/icons/Search";
import VisibilityIcon from "@material-ui/icons/Visibility";
import {
  MuiPickersUtilsProvider,
  KeyboardDatePicker,
} from "@material-ui/pickers";
```

```
import DateFnsUtils from "@date-io/date-fns";
import Pagination from '@material-ui/lab/Pagination';

import React from "react";
import Page from "src/component/Page";
// import SearchFilter from "../../../component/SearchFilter";
import { usePagination } from "@material-ui/lab/Pagination";
const useStyles = makeStyles((theme) => ({
  btn: {
    width: "100%",
    backgroundColor: "black",
    borderBottomLeftRadius: 5,
    borderBottomRightRadius: 5,
  },
  table: {
    minWidth: 650,
  },
  ul: {
    listStyle: "none",
    padding: 0,
    margin: 0,
    display: "flex",
  },
  formControl: {
    margin: theme.spacing(1),
    minWidth: 120,
  },
}));
function createData(
  Sr_No,
  UserName,
  Earning_Mode,
  Current_balance,
  Product_Name,
  Total_Earning,
  Units,
  Last_Payout_Date_and_Time,
  Actions
) {
  return {
    Sr_No,
    UserName,
    Earning_Mode,
    Current_balance,
    Product_Name,
    Total_Earning,
    Units,
    Last_Payout_Date_and_Time,
    Actions,
  };
}

const rows = [
  createData(
  1,
    "Rose",
    "Wallet",
    1000,
    "Dinning-tabel",
    5000,
    2,
    "25/03/2021,5:16 Pm",
    5
  ),
  createData(
    2,
    "sheetal",
    "Investmen",
    1000,
    "Almira",
    2000,
    3,
    "21/03/2021 3:30 PM"
  ),
  createData(
    3,
    "sheetal",
    "Investmen",
    1000,
    "Almira",
    2000,
    3,
    "21/03/2021 3:30 PM"
  ),
  createData(
    4,
    "sheetal",
    "Investmen",
    1000,
    "Almira",
    2000,
    3,
    "21/03/2021 3:30 PM"
  ),
  createData(
    5,
    "sheetal",
    "Investmen",
    1000,
    "Almira",
    2000,
    3,
    "21/03/2021 3:30 PM"
  ),
  createData(
    6,
    "sheetal",
    "Investmen",
    1000,
    "Almira",
    2000,
    3,
    "21/03/2021 3:30 PM"
  ),
  createData(
    7,
    "sheetal",
    "Investmen",
    1000,
    "Almira",
    2000,
    3,
    "21/03/2021 3:30 PM"
  ),
];
export default function (props) {
  const Classes =  useStyles();
  const { items } = usePagination({
  count: 10,
  });
  const [state, setState] = React.useState({
    age: "",
    name: "hai",
  });
  return (
  <Container maxWidth="xl">
    <Page title="user-management">
      <Box
        container
        maxWidth="lg"
        style={{ height: 1000, backgroundColor: "white" }}
      >
        <Box
          py={2}
          borderBottom={1}
          style={{ borderColor: "#cccccc", margin: 15 }}
        >
          <Typography
            style={{ fontSize: 18, fontWeight: "bold", color: "#343a40" }}
          >
            EARNING MANAGEMENT
          </Typography>
        </Box>
        <Divider />
        <Box pb={2}>
          <SearchFilter />
        </Box>

        <TableContainer component={Paper}>
          <Table className={Classes.table} aria-label="simple table">
            <TableHead>
              <TableRow>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                >
                  Sr.No
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  UserName
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  Earning Mode
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  Current balance
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  Product Name
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  Total Earning
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  Units
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="right"
                >
                  Last payout Date and Time
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Action
                </TableCell>
              </TableRow>
            </TableHead>
            <TableBody>
              {rows.map((row) => (
                <TableRow key={row.name}>
                  <TableCell component="th" scope="row">
                    {row.Sr_No}
                  </TableCell>
                  <TableCell align="center">{row.UserName}</TableCell>
                  <TableCell align="center">{row.Earning_Mode}</TableCell>
```

```
                    <TableCell align="center">{row.Current_balance}</TableCell>
                    <TableCell align="center">{row.Product_Name}</TableCell>
                    <TableCell align="center">{row.Total_Earning}</TableCell>
                    <TableCell align="center">{row.Units}</TableCell>
                    <TableCell align="center">
                      {row.Last_Payout_Date_and_Time}
                    </TableCell>
                    <TableCell style={{ width: 5 }} align="right">
                      <Link href="/EarningView">
                        <Button
                          variant="contained"
                          color="primary"
                          style={{
                            minWidth: "initial",
                            padding: "6px",
                            marginLeft: "7px",
                          }}
                        >
                          <VisibilityIcon style={{ fontSize: "15px" }} />
                        </Button>
                      </Link>
                    </TableCell>
                  </TableRow>
                ))}
              </TableBody>
            </Table>

          </TableContainer>
          <Box display="flex" justifyContent="flex-end">
          <Pagination count={10} shape="rounded" />
          </Box>
        </Box>
      </Page>
    </Container>
  );
}
const SearchFilter = ({ exportProps, addProps }) => {
  const [selectedDate, setSelectedDate] = React.useState(
  new Date("2014-08-18T21:11:54")
  );

  const handleDateChange = (date) => {
    setSelectedDate(date);
  };
```

```
return (
  <Card raised>
    <CardContent>
      <Typography variant="h4" style={{ marginBottom: "8px",paddingLeft:20 }}>
        Filter by
      </Typography>

      <Grid spacing={2} container alignItems="center">
        <Grid item md={2}>
          <Box style={{ marginBottom: 10,marginLeft:20 }}>
            <Typography style={{ paddingBottom: 10 }} variant="h5">
              Search by:
            </Typography>

            <Grid item md={2}>
          <Box style={{ marginTop: 10,width:210 }}>
            <FormControl variant="outlined">
              <OutlinedInput
                placeholder="search by name"
                endAdornment={
                  <InputAdornment position="end">
                    <SearchIcon
                      aria-label="toggle password visibility"
                      edge="end"
                    ></SearchIcon>
                  </InputAdornment>
                }
              />
            </FormControl>
          </Box>
        </Grid>
          </Box>
        </Grid>
        <Grid item md={2}>
          <Box>
            <Typography variant="h5" style={{paddingLeft:55}}>From Date</Typography>
            <MuiPickersUtilsProvider utils={DateFnsUtils}>
              <KeyboardDatePicker
              style={{marginLeft:55,width:170}}
              inputVariant="outlined"
              disableToolbar
                variant="inline"
                format="MM/dd/yyyy"
                margin="normal"
                id="date-picker-inline"
                value={selectedDate}
                onChange={handleDateChange}
                KeyboardButtonProps={{
                  "aria-label": "change date",
                }}
              />
            </MuiPickersUtilsProvider>
          </Box>
        </Grid>
        <Grid item md={2}>
          <Box>
            <Typography variant="h5" style={{paddingLeft:55}}>To Date</Typography>
            <MuiPickersUtilsProvider utils={DateFnsUtils}>
              <KeyboardDatePicker
              style={{paddingLeft:55,width:170}}
              inputVariant="outlined"
              disableToolbar
                variant="inline"
                format="MM/dd/yyyy"
                margin="normal"
                id="date-picker-inline"
                value={selectedDate}
                onChange={handleDateChange}
                KeyboardButtonProps={{
                  "aria-label": "change date",
                }}
              />
            </MuiPickersUtilsProvider>
          </Box>
        </Grid>
        <Grid item>
          <Box style={{ paddingTop: 48,paddingLeft:55 }}>
            <Button variant="contained" color="primary">
              Search
            </Button>
            <Button
              variant="contained"
              color="primary"
              style={{ marginLeft: "3px", marginRight: "3px" }}
            >
              Reset
            </Button>
          </Box>
          {exportProps && (
            <Button
```

```
            variant="contained"
            color="primary"
            onClick={exportProps.onClick}
          >
            {exportProps.title}
          </Button>
        )}
      </Grid>
    </Grid>

    <Box display="flex" justifyContent="flex-end" py={4}>
      {addProps && (
        <Button variant="contained" color="primary">
          {addProps.title}
        </Button>
      )}
    </Box>
  </CardContent>
</Card>
 );
};
```

All Assets

```
import React from "react";
import {
 Container,
 Divider,
 Box,
 Card,
 Grid,
 CardContent,
 Typography,
 Button,
 Link,
} from "@material-ui/core";
import Page from "src/component/Page";

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 EarningView = () => {
  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 Earning</strong>
            </Typography>
            <Divider />
          </Box>

          <Card>
            <CardContent>
              <Grid container md={6} direction="column" spacing={3} style={{paddingBottom:'50p
x'}}>
                <Grid item>
                  {" "}
                  <Row field="Earning Mode :" value="Wallet" />
                </Grid>
                <Grid item>
                  {" "}
                  <Row field="Current Balance :" value="  1000" />
                </Grid>
                <Grid item>
                  {" "}
                  <Row field="Product Name :" value="Dinning-tabel" />
                </Grid>
                <Grid item>
                  {" "}
                  <Row field="Total Earning :" value="5000" />
                </Grid>
                <Grid item>
                  <Row field="Units :" value="2" />
                </Grid>
                <Grid item >
                  <Row field="Last Payout Date and Time :" value="20/01/2021,9:30 AM" />
                </Grid>


                <Grid item>
                    <Link href='/EarningManagement'>
                    <Button variant='contained' color='primary' size='large'>Close</Button>
                    </Link>

                </Grid>
              </Grid>
            </CardContent>
          </Card>
        </Page>
      </Container>
    </>
  );
};

export default EarningView;
```

All Trancations

```
import React, { useState } from "react";
import SearchFilter from "../../../component/SearchFilter";
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 { 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",
    ),
  ];
```

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

const tabChange = (event, tabName) => {
  setTab(tabName);
};

//  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: "" }}
            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">
       <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>
    {rows.map((row) => (
      <TableRow key={row.name}>
        <TableCell component="th" scope="row">
          {row.Sr_No}
        </TableCell>
        <TableCell align="left">{row.LegalName}</TableCell>
        <TableCell align="left">{row.Document_Id_Type}</TableCell>
        <TableCell align="left">{row.Document_Id_Number}</TableCell>
        <TableCell align="left">{row.Country}</TableCell>
        <TableCell align="left">{row.Status}</TableCell>

        <TableCell style={{ width: 5 }} align="left">
          <Box display="flex">
          <Link href="/viewKYC">
  <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>
 );
}
```

Transcation Details

```
import React 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";

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">{field}</Typography>:
      </Box>
    </Grid>
    <Grid item md={6}>
      <Typography variant="body1">{value}</Typography>
    </Grid>
  </Grid>
);
const ViewKYC = () => {
  const classes = useStyles();
  const [isApprove, setApprove] = React.useState(false);

  const openAprove = () => {
    setApprove(true);
  };

  const closeApprove = () => {
    setApprove(false);
  };

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

  const openReject = () => {
    setReject(true);
  };

  const closeReject = () => {
    setReject(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="GH1153PL28" />
                <Row field="KYC Status" value="Approved" />
                <Row field="Created At" value="02/08/2020, 5:30 PM" />
                <Row field="Type" value="eKYC" />
                <Row field="Id Type" value="Govt. ID" />
                <Row field="Id Number" value="PAN1234" />
              </Grid>
              <Grid container item md={6}>
                <Row field="Company Legal Name" value="Bharat Singh" />
                <Row field="Trade Licence Number" value="1564 5523 1236" />
                <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={PanFront} alt="front" sizes='small' className={classes.images}/>{" "}
              <Typography variant="h2">Front</Typography>
            </Grid>
            <Grid item md={4}>
              <img src={PanRear} alt="front" sizes='small'  className={classes.images}/>
              <Typography variant="h2">Rear</Typography>
            </Grid>
            <Grid item md={4} >
              <img src={Selfie} alt="front" width='260px'  className={classes.images}/>
              <Typography variant="h2">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">
      Are you sure you want to approve this document?
    </DialogContentText>
  </DialogContent>
  <DialogActions>
    <Button color="primary">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">
              Are you sure you want to reject this document?
            </DialogContentText>

            <TextField
              fullWidth
              placeholder="Specify reason fo rejection"
            />
          </DialogContent>
          <DialogActions>
            <Button color="primary">Yes</Button>
            <Button onClick={closeReject} color="primary" autoFocus>
              No
            </Button>
          </DialogActions>
        </Dialog>
      </Container>
    </Box>
   </Container>
   </Page>
  </>
 );
};

export default ViewKYC;
```

Trancation Type

```
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,
} from "@material-ui/core";
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 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>
              </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>
                  </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);
        auth.userLogIn(true, response.data.result.token);
        // history.push("/otp",email);
        history.push({
          // pathname: '/otp',
          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" }}>
  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={3}>
  <Typography variant="h3">From Date</Typography>
  <MuiPickersUtilsProvider utils={DateFnsUtils}>
    <KeyboardDatePicker
      inputVariant="outlined"
      disableToolbar
      variant="inline"
      format="MM/dd/yyyy"
      margin="normal"
      id="date-picker-inline"
      value={selectedDate}
      onChange={handleDateChange}
      KeyboardButtonProps={{
        "aria-label": "change date",
      }}
      size="small"
    />
  </MuiPickersUtilsProvider>
</Grid>
<Grid item md={3}>
  {" "}
  <Typography variant="h3">To Date</Typography>
  <MuiPickersUtilsProvider utils={DateFnsUtils}>
    <KeyboardDatePicker
      inputVariant="outlined"
      disableToolbar
      variant="inline"
      format="MM/dd/yyyy"
      margin="normal"
      id="date-picker-inline"
      value={selectedDate}
      onChange={handleDateChange}
      KeyboardButtonProps={{
        "aria-label": "change date",
      }}
      size="small"
    />
  </MuiPickersUtilsProvider>
</Grid>
{searchProps && (
  <Grid item md={3}>
    <Box mt={4}>
      <TextField
        variant="outlined"
        placeholder={searchProps}
```

```
          size="small"
        />
      </Box>
    </Grid>
  )}
  <Grid item md={3}>
    <Box
      style={{ height: "65px" }}
      display="flex"
      alignItems="flex-end"
    >
      <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"
                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>
);
};
```

Review Withdrawl

```
import React, { useRef } from "react";
import { makeStyles } from "@material-ui/core/styles";

// import { Editor } from "@tinymce/tinymce-react";
import {
  Typography,
  Box,
  Divider,
  TextField,
  Button,
} from "@material-ui/core";
import Page from "src/component/Page";
// import classes from "*.module.css";

const useStyles = makeStyles({
  table: {
    // minWidth: 700,
  },
  button_style: {
    width: 130,
  },
});

export default function (props) {
  const classes = useStyles();
  const editorRef = useRef(null);
  const log = () => {
    if (editorRef.current) {
      console.log(editorRef.current.getContent());
    }
  };
  return (
    <Page title="user-management">
      <Box style={{ padding: 20 }}>
        {/* <Typography variant={"h3"}>PRIVACY POLICY</Typography>
        <Divider style={{ marginTop: 10 }}></Divider> */}
        <Typography variant="h3" style={{ marginBottom: "8px" }}>
          <strong>PRIVACY POLICY</strong>
        </Typography>
      <Divider />
     <Box
      display="flex"
      justifyContent="center"
      // bgcolor='red'
     >
      <Box
       my={7}
       boxShadow={10}
       borderRadius={5}
       bgcolor="#fff"
       // height={300}
       // paddingBottom={}
       padding={3}
       width="80%"
      >
       <Box>
        <Box display="flex">
         <Box display="flex" width="20%" alignItems="center" justifyContent="space-
between">
          <Typography>Page Name</Typography>
          <span>:</span>
         </Box>
         <Box ml={10} width="70%">
          <TextField
           placeholder="Privacy Policy"
           aria-readonly
           variant="outlined"
           style={{ width: "100%" }}
           // placeholder="what is KYC?"
          />
         </Box>
        </Box>
        <Box my={3}>
         <Box display="flex" width="20%" justifyContent="space-between">
          <Typography>Description</Typography>
          <span>:</span>
         </Box>
         <Box width="100%" minHeight={100} my={2}>
          {/* <Editor
           onInit={(evt, editor) => (editorRef.current = editor)}
           init={{
            height: 500,
            menubar: false,
            plugins: [
                  "advlist autolink lists link image charmap print preview anchor",
                  "searchreplace visualblocks code fullscreen",
                  "insertdatetime media table paste code help wordcount",
                ],
                toolbar:
                  "undo redo | formatselect | " +
                  "bold italic backcolor | alignleft aligncenter " +
                  "alignright alignjustify | bullist numlist outdent indent | " +
                  "removeformat | help",
                content_style:
                  "body { font-family:Helvetica,Arial,sans-serif; font-size:14px }",
              }}
            /> */}
            <Box display='flex' my={5} justifyContent='center'>
            <Button
              onClick={log}
              variant="contained"
              color="primary"
              className={(classes.button_style)}
            >
              Update
            </Button>
            </Box>
          </Box>
        </Box>
      </Box>
    </Box>
  </Box>
 </Box>
 </Page>
 );
}
```

WithdrawII

```
import React, { useState, useEffect } from "react";
import {
  Container,
  Divider,
  Box,
  Card,
  Grid,
  CardContent,
  Typography,
  Button,
  Link,
  TextField,
  MenuItem,
  FormControl,
} from "@material-ui/core";
import Page from "src/component/Page";
import {
  isValidAlphabet,

  // isValidContact,
  isValidProductPrice,
  isValidMetric,
  isValidValueOfPurchase,
  isValidProductType,
  isValidPayout,
  isValidAlNumber,
} from "../../../Validation/Validation";
import { useHistory } from "react-router-dom";
import axios from "axios";
import ApiConfig from "src/config/APIConfig";
const Units = [
  {
    value: "kg",
    label: "Kg",
  },
  {
    value: "Unit",
    label: "Unit",
  },
  {
    value: "punce",
    label: "Ounce",
  },
];
const Currencies = [
  {
    value: "USD",
    label: "USD",
  },
  {
    value: "GBP",
    label: "GBP",
  },

];
const Frequencies = [
  {
    value: "Monthly",
    label: "Monthly",
  },
  {
    value: "Quarterly",
    label: "Quarterly",
  },
  {
    value: "Annually",
    label: "Annually",
  },

];


const EditProduct = () => {
const[prodImg,setProductImgBuild]=useState(null)
const[baseImg,setBaseImage]=useState(null)
  const [unit, setUnit] = React.useState("units");
  const [frequency, setFrequency] = React.useState("units");
 const [currency, setCurrency] = React.useState("USD");
 const [price, setPrice] = React.useState("");
 const[img,setImg]=useState()
  const handleChange = (event) => {
    setUnit(event.target.value);
  };
  const handleCurrency = (event) => {
    setCurrency(event.target.value);
  };
  const handleFrequency = (event) => {
    setFrequency(event.target.value);
  };
  const history = useHistory();
  const [number, setNumber] = useState(false);
  const [name, setName] = useState(true);
  const [err, setErr] = useState(false);
  const [isSubmit, setIsSubmit] = useState(false);
  const accessToken = window.localStorage.getItem("creatturAccessToken");
  const [formData, setFormData] = useState({
```

```
  // mobileNumber: "",
  name: "",
  metric: "",
  productPrice: "",
  valueOfPurchase: "",
  productType: "",
  currency:currency,
  payout:frequency,
  price: "",
  interest: "",
  image:baseImg
  // lastName: "",
  // stateName: "",
  // cityName: "",
});
const _onInputChange = (e) => {
  const name = e.target.name;
  const value = e.target.value;
  const temp = { ...formData, [name]: value };
  setFormData(temp);
};
const getBase64 = (file, cb) => {
  let reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onload = function () {
  cb(reader.result);
  };
  reader.onerror = function (err) {
    console.log('Error: ', err);
  };
};

const _onProfilePicChange = (e) => {
  console.log('eeee', e);
  const name = e.target.name;
  const value = URL.createObjectURL(e.target.files[0]);
  setProductImgBuild(value); //will give displayable image use it for preview also
  getBase64(e.target.files[0], (result) => {
    console.log('result', result);
    setBaseImage(result); //will give base64
    const temp = { ...formData, [name]: result };
    console.log('temp', temp);
    setFormData(temp);
  });
```

```
};
const addProduct = () => {
  console.log('hasgy')
  axios
    .post('http://182.72.203.245:1829/api/v1/admin/addProduct', {
      productName: formData.name,
      price: formData.productPrice,
      metric: unit,
      currencyType: formData.currency,
      minimumValueOfPurchase: formData.valueOfPurchase,
      interest: formData.interest,
      payoutFrequency: formData.payout,
      image: formData.image,
    },{
      headers: {
        token: accessToken,
      },
    })

    .then((response) => {
      if (response.data.status !== 200) {
        // setIserror(true);
        // setIsSuccess(false);
        // setAlertMsg(response.data.message);
      } else {
       console.log(response)
        //  else setHistory(withdrawFilter);
      }
      // setIsLoading(false);
    })
    .catch((response) => {
      // setIsUpdating(false);

      console.log("response", response);
    });
};
const onFileChange=(e)=>{
  setImg(e.target.files[0] )
  // formData.image :
}
const submitHandler = () => {
  setIsSubmit(true);

  if (
    isValidAlphabet(formData.name) &&
    // isValidContact(formData.mobileNumber) &&
    isValidMetric(formData.metric) &&
    isValidProductPrice(formData.productPrice) &&
    isValidValueOfPurchase(formData.valueOfPurchase) &&
    isValidProductType(formData.productType) &&
    isValidAlNumber(formData.price) &&
    isValidPayout(formData.payout)
  ) {
    // saveFormData(formData);

    addProduct();
  }
};
return (
  <>
    <Container maxWidth="xl">
      <Page
        style={{ display: "flex", flexDirection: "column" }}
        title="Update product"
      >
        <Box pt={3} mb={8}>
          <Typography variant="h3" style={{ marginBottom: "8px" }}>
            <strong>ADD NEW PRODUCT</strong>
          </Typography>
          <Divider />
        </Box>

        <Card>
          <CardContent>
            <Box mt={2}>
              <FormControl>
                <Grid container spacing={5} style={{ marginBottom: "26px" }}>
                  <Grid item md={12}>
                    <Typography
                      variant="h4"
                      style={{ fontWeight: "bold", marginBottom: "12px" }}
                    >
                      Enter Product Name
                    </Typography>
                    <TextField
                      variant="outlined"
                      fullWidth
                      placeholder="Enter Product Name"
                      required
                      name="name"
              value={formData.name}
              onChange={_onInputChange}
              error={
                formData.name !== "" &&
                !isValidAlphabet(formData.name)
              }
              helperText={
                formData.name !== "" &&
                !isValidAlphabet(formData.name) &&
                "please enter valid product name"
              }
            />
          </Grid>
          <Grid item md={12}>
            <Typography
              variant="h4"
              style={{ fontWeight: "bold", marginBottom: "12px" }}
            >
              Metric
            </Typography>
            <TextField
              variant="outlined"
              placeholder="Enter Metric"
              style={{ width: "49%", marginRight: "2%" }}
              name="metric"
              value={formData.metric}
              // onChange={mobileHandler}
              onChange={_onInputChange}
              error={
                formData.metric !== "" &&
                !isValidMetric(formData.metric)
              }
              helperText={
                formData.metric !== "" &&
                !isValidMetric(formData.metric) &&
                "please enter valid metric"
              }
            />
            <TextField
              variant="outlined"
              select
              label="Select"
              value={unit}
              onChange={handleChange}
              style={{ width: "49%" }}
          >
            {Units.map((option) => (
              <MenuItem key={option.value} value={option.value}>
                {option.label}
              </MenuItem>
            ))}
          </TextField>
        </Grid>

        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Enter Product Price
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            type="number"
            placeholder="Enter Product Price"
            name="productPrice"
            value={formData.productPrice}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.productPrice !== "" &&
              !isValidProductPrice(formData.productPrice)
            }
            helperText={
              formData.productPrice !== "" &&
              !isValidProductPrice(formData.productPrice) &&
              "please enter valid product price"
            }
          />
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Enter Currency
          </Typography>
          <TextField
            variant="outlined"
            select
            label="Select"
            value={currency}
            onChange={handleCurrency}
            style={{ width: "49%" }}
          >
            {Currencies.map((option) => (
              <MenuItem key={option.value} value={option.value}>
                {option.label}
              </MenuItem>
            ))}
          </TextField>

        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            {" "}
            Product Type
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            placeholder="Computer"
            name="productType"
            value={formData.productType}
            onChange={_onInputChange}
            error={
              formData.productType !== "" &&
              !isValidProductType(formData.productType)
            }
            helperText={
              formData.productType !== "" &&
              !isValidProductType(formData.productType) &&
              "please enter valid product type"
            }
          />
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
```

```
            Enter Minimum Value of Purchase{" "}
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            placeholder="Enter Value Of Purchase"
            name="valueOfPurchase"
            value={formData.valueOfPurchase}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.valueOfPurchase !== "" &&
              !isValidValueOfPurchase(formData.valueOfPurchase)
            }
            helperText={
              formData.valueOfPurchase !== "" &&
              !isValidValueOfPurchase(formData.valueOfPurchase) &&
              "please enter valid value of purchase"
            }
          />
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Enter Product Payout Frequency{" "}
          </Typography>
          <TextField
            variant="outlined"
            select
            label="Select"
            value={frequency}
            onChange={handleFrequency}
            style={{ width: "49%" }}
          >
            {Frequencies.map((option) => (
              <MenuItem key={option.value} value={option.value}>
                {option.label}
              </MenuItem>
            ))}
          </TextField>
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
```

```
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Interest{" "}
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            type="number"
            placeholder="Interest"
            name="interest"
            value={formData.interest}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.interest !== "" &&
              !isValidPayout(formData.interest)
            }
            helperText={
              formData.interest !== "" &&
              !isValidPayout(formData.interest) &&
              "please enter valid interest value "
            }
          />
        </Grid>

        <Grid item md={12}>
        <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}

          >
            Add Image{" "}
          </Typography>
          <Button variant="contained" color='primary' component="label"> Upl
oad File <input name='image' onChange={_onProfilePicChange} variant='contained'
type="file"  hidden/>
          </Button>

            {prodImg &&  <img src={prodImg} alt='product' width='10%'/>}

        </Grid>
        <Grid item md={12}>
          <Link href="/product">
            <Button
```

```
                  variant="contained"
                  color="primary"
                  size="large"
                  style={{ marginRight: "8px" }}
                >
                  Cancel
                </Button>
              </Link>
              {/* <Link href="/product"> */}
              <Button
                variant="contained"
                color="primary"
                size="large"
                onClick={submitHandler}
              >
                Submit
              </Button>
              {/* </Link> */}
            </Grid>
          </Grid>
        </FormControl>
      </Box>
    </CardContent>
  </Card>
 </Page>
</Container>
</>
 );
};

export default EditProduct;
```

Depostie Coins

```
import React,{useState} from "react";
import {
  Container,
  Divider,
  Box,
  Card,
  Grid,
  CardContent,
  Typography,
  Button,
  Link,
  TextField,
  MenuItem,
  FormControl,
} from "@material-ui/core";
import Page from "src/component/Page";
import {
  isValidAlphabet,

  // isValidContact,
  isValidProductPrice,
  isValidMetric,
  isValidValueOfPurchase,
  isValidProductType,
  isValidPayout
} from "../../../Validation/Validation";
import { useHistory } from "react-router-dom";
const Units = [
  {
    value: "USD",
    label: "Kg",
  },
  {
    value: "EUR",
    label: "Units",
  },
  {
    value: "BTC",
    label: "Ounce",
  },
];
const EditProduct = () => {
  const [unit, setUnit] = React.useState("EUR");

  const handleChange = (event) => {
    setUnit(event.target.value);
  };
  const history = useHistory();
  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: "",
  name: "",
  metric:"",
  productPrice:"",
  valueOfPurchase:"",
  productType:"",
  payout:"",
  // lastName: "",
  // stateName: "",
  // cityName: "",
});
const _onInputChange = (e) => {
  const name = e.target.name;
  const value = e.target.value;
  const temp = { ...formData, [name]: value };
  setFormData(temp);
};

const submitHandler = () => {
  setIsSubmit(true);

  if (
    isValidAlphabet(formData.name) &&

    // isValidContact(formData.mobileNumber) &&
    isValidMetric(formData.metric) &&
    isValidProductPrice(formData.productPrice) &&
    isValidValueOfPurchase(formData.valueOfPurchase) &&
    isValidProductType(formData.productType) &&
    isValidPayout(formData.payout)
  )
   {
    // saveFormData(formData);

    history.push("/product");
  }
};
return (
  <>
    <Container maxWidth="xl">
      <Page
        style={{ display: "flex", flexDirection: "column" }}
        title="Update product"
      >
```

```
<Box pt={3} mb={8}>
  <Typography variant="h3" style={{ marginBottom: "8px" }}>
    <strong>Update product</strong>
  </Typography>
  <Divider />
</Box>

<Card>
  <CardContent>
    <Box mt={2}>
      <FormControl>
        <Grid container spacing={5} style={{ marginBottom: "26px" }}>
          <Grid item md={12}>
            <Typography
              variant="h4"
              style={{ fontWeight: "bold", marginBottom: "12px" }}
            >
              Enter Product Name
            </Typography>
            <TextField
              variant="outlined"
              fullWidth
              placeholder="Enter Product Name"
              required
              name="name"
              value={formData.name}
    onChange={_onInputChange}
    error={formData.name !== "" && !isValidAlphabet(formData.name)}
    helperText={
      formData.name !== "" &&
      !isValidAlphabet(formData.name) &&
      "please enter valid product name"
    }
            />
          </Grid>
          <Grid item md={12}>
            <Typography
              variant="h4"
              style={{ fontWeight: "bold", marginBottom: "12px" }}
            >
              Metric
            </Typography>
            <TextField
              variant="outlined"
              placeholder="Enter Metric"
            style={{ width: "49%", marginRight: "2%" }}
            name="metric"
            value={formData.metric}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.metric !== "" &&
              !isValidMetric(formData.metric)
            }
            helperText={
              formData.metric !== "" &&
              !isValidMetric(formData.metric) &&
              "please enter valid metric"
            }
          />
          <TextField
            variant="outlined"
            select
            label="Select"
            value={unit}
            onChange={handleChange}
            style={{ width: "49%" }}
          >
            {Units.map((option) => (
              <MenuItem key={option.value} value={option.value}>
                {option.label}
              </MenuItem>
            ))}
          </TextField>
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Enter Product Price
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            placeholder="Enter Product Price"
            name="productPrice"
            value={formData.productPrice}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.productPrice !== "" &&
              !isValidProductPrice(formData.productPrice)
            }
            helperText={
              formData.productPrice !== "" &&
              !isValidProductPrice(formData.productPrice) &&
              "please enter valid product price"
            }
          />
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            {" "}
            Product Type
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            placeholder="Currency"
            name="productType"
            value={formData.productType}
      onChange={_onInputChange}
      error={formData.productType !== "" && !isValidProductType(formData.pr
oductType)}
      helperText={
        formData.productType !== "" &&
        !isValidProductType(formData.productType) &&
        "please enter valid product type"
      }
          />
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Enter Min Value of Purchase{" "}
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            placeholder="Enter Value of purchase"
            name="valueOfPurchase"
            value={formData.valueOfPurchase}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.valueOfPurchase !== "" &&
              !isValidValueOfPurchase(formData.valueOfPurchase)
            }
            helperText={
              formData.valueOfPurchase !== "" &&
              !isValidValueOfPurchase(formData.valueOfPurchase) &&
              "please enter valid minimum value of purchase"
            }
          />
        </Grid>
        <Grid item md={6} xl={6} sm={12} xs={12}>
          <Typography
            variant="h4"
            style={{ fontWeight: "bold", marginBottom: "12px" }}
          >
            Enter Product Payout{" "}
          </Typography>
          <TextField
            variant="outlined"
            fullWidth
            placeholder="Quaterly"
            name="payout"
            value={formData.payout}
            // onChange={mobileHandler}
            onChange={_onInputChange}
            error={
              formData.payout !== "" &&
              !isValidPayout(formData.payout)
            }
            helperText={
              formData.payout !== "" &&
              !isValidPayout(formData.payout) &&
              "please enter valid payout "
            }
          />
        </Grid>

        <Grid item md={12}>
          <Typography
```

```
                        variant="h4"
                        style={{ fontWeight: "bold", marginBottom: "12px" }}
                      >
                        Upload Image{" "}
                      </Typography>
                     <Button variant='contained'>Choose File</Button>
                    </Grid>
                    <Grid item md={12}>
                      <Link href="/product">
                      <Button
                        variant="contained"
                        color="primary"
                        size="large"
                        style={{ marginRight: "8px" }}
                      >
                        Cancel
                      </Button>
                      </Link>
                        <Button
                          variant="contained"
                          color="primary"
                          size="large"
                          onClick={submitHandler}

                        >
                          Submit
                        </Button>

                    </Grid>
                  </Grid>
                </FormControl>
              </Box>
            </CardContent>
          </Card>
        </Page>
      </Container>
    </>
  );
};

export default EditProduct;
```

```
import React,{useEffect} from "react";
import SearchFilter from "../../../component/SearchFilter";
import computerImage from "../../../images/pc.jpeg";
import {
  Container,
  Divider,
  Box,
  Typography,
  Button,
  Dialog,
  DialogActions,
  DialogContent,
  DialogContentText,
  Link,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  TableBody,
  Table,
  Paper,

} from "@material-ui/core";
import Pagination from '@material-ui/lab/Pagination';
import axios from "axios";
import ApiConfig from "src/config/APIConfig";
import BlockIcon from "@material-ui/icons/Block";
import DeleteIcon from "@material-ui/icons/Delete";
import EditIcon from "@material-ui/icons/Edit";
import VisibilityIcon from "@material-ui/icons/Visibility";

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

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 accessToken = window.localStorage.getItem("creatturAccessToken");
  const classes = useStyles();
  const [isBlock, setBlock] = React.useState(false);

  const openBlock = () => {
    setBlock(true);
  };

  const closeBlock = () => {
    setBlock(false);
  };

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

  const openDelete = () => {
    setDelete(true);
  };

  const closeDelete = () => {
    setDelete(false);
  };

  useEffect(() => {
    // setIsLoading(true);
    axios.post(ApiConfig.getProductList,{}, {

       headers:{
        token:accessToken,
       }

      })
      .then((response) => {
        if (response.data.responseCode !== 200) {

        } else {
          // setDisplayname
          // console.log(result)
          console.log(response)

          // else setHistory(depositFilter);

        }
        // setIsLoading(false);
      })
      .catch((response) => {
        // setIsUpdating(false);

        console.log("response", response);
      });
  }, []);
  function createData(
    Product_Id,
    Product_Image,
    Product_Name,
    Metric,
    Price,
    Currency_Type,
    Min_Value_Of_Purchase,
    Payout_Frequency,
    Actions
  ) {
    return {
      Product_Id,
      Product_Image,
      Product_Name,
      Metric,
      Price,
      Currency_Type,
      Min_Value_Of_Purchase,
      Payout_Frequency,
      Actions,
    };
  }
  const rows = [
    createData(1, " ","Computer","10 Units", "$400", "Currency", 2, "weekly"),
    createData(2," ", "Computer", "10 Units", "$400", "Currency", 2, "weekly"),

    createData(3," ", "Computer", "10 Units", "$400", "Currency", 2, "weekly"),
    createData(4," ", "Computer", "10 Units", "$400", "Currency", 2, "weekly",),
    createData(5," ", "Computer", "10 Units", "$400", "Currency", 2, "weekly",),

    createData(6," ", "Computer", "10 Units", "$400", "Currency", 2, "weekly",),

  ];

  return (
    <Container
      maxWidth="xl"
      style={{ overflowX: "auto", paddingBottom: "100px" }}
    >
      <Page
        style={{ display: "flex", flexDirection: "column" }}
        title="Product Management"
      >
        <Box py={3}>
          <Typography variant="h3" style={{ marginBottom: "8px" }}>
            <strong>PRODUCT MANAGEMENT</strong>
          </Typography>
          <Divider />

          <Box py={5}>
            <SearchFilter
              searchProps="Search By Product Name"
              addProps={{ title: "add new", link: "/addProduct" }}
            />
          </Box>
        </Box>

        <div style={{ overflow: "auto" }}>
          {/* <DataGrid
            className={classes.pdbt}
            rows={rows}
            columns={columns}
            pageSize={5}
            autoHeight
          /> */}
          <TableContainer component={Paper}>
            <Table className={classes.table} aria-label="simple table">
              <TableHead>
                <TableRow>
                  <TableCell
                    style={{ color: "white", backgroundColor: "#252d47" }}
                  >
                    Product Id
                </TableCell>
                <TableCell
                  align="center"
                  style={{ color: "white", backgroundColor: "#252d47" }}
                >
                  Product Image
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Product Name
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Metric
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Price
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Currency Type
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Min Value Of Purchase
                </TableCell>
                <TableCell
                  style={{ color: "white", backgroundColor: "#252d47" }}
                  align="center"
                >
                  Payout Frequency
                </TableCell>

                <TableCell
                style={{ color: "white", backgroundColor: "#252d47" }}
                align="center"
              >
                Actions
              </TableCell>
            </TableRow>
          </TableHead>
          <TableBody>
            {rows.map((row) => (
              <TableRow key={row.name}>
                <TableCell component="th" scope="row">
                  {row.Product_Id}
                </TableCell>
                {/* <TableCell align="center">{row.Product_Image}</TableCell> */}
                <TableCell style={{ width: 5 }} align="center">
                  <Box display="flex">
                    <img src={computerImage} alt="computer" width="40px" />
                  </Box>
                </TableCell>
                <TableCell align="center">{row.Product_Name}</TableCell>
                <TableCell align="center">{row.Metric}</TableCell>
                <TableCell align="center">{row.Price}</TableCell>
                <TableCell align="center">{row.Currency_Type}</TableCell>
                <TableCell align="center">
                  {row.Min_Value_Of_Purchase}
                </TableCell>
                <TableCell align="center">{row.Payout_Frequency}</TableCell>

                <TableCell style={{ width: 5 }} align="right">
                  <Box display="flex">
                    <Link href="/viewProduct">
                      <Button
                        variant="contained"
                        className={classes.button}
                      >
                        <VisibilityIcon style={{ fontSize: "15px" }} />
                      </Button>
                    </Link>

                    <Link href="/editProduct">
                      <Button
                        variant="contained"
                        className={classes.button}
                      >
                        <EditIcon style={{ fontSize: "15px" }} />
                </Button>
              </Link>

              <Button
                variant="contained"
                color="primary"
                className={classes.button}
                onClick={openBlock}
              >
                <BlockIcon style={{ fontSize: "15px" }} />
              </Button>
              <Button
                variant="contained"
                color="secondary"
                className={classes.button}
                onClick={openDelete}
              >
                <DeleteIcon style={{ fontSize: "15px" }} />
              </Button>
            </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>
</div>

<Dialog
  open={isBlock}
  onClose={closeBlock}
  aria-labelledby="alert-dialog-title"
  aria-describedby="alert-dialog-description"
>
  <DialogContent>
    <DialogContentText id="alert-dialog-description">
      Are you sure you want to block this product??
    </DialogContentText>
  </DialogContent>
  <DialogActions>
    <Button color="primary">Yes</Button>
    <Button onClick={closeBlock} color="primary" autoFocus>
      No
    </Button>
  </DialogActions>
</Dialog>

<Dialog
  open={isDelete}
  onClose={closeDelete}
  aria-labelledby="alert-dialog-title"
  aria-describedby="alert-dialog-description"
>
  <DialogContent>
    <DialogContentText id="alert-dialog-description">
      Are you sure you want to delete this product?
    </DialogContentText>
  </DialogContent>
  <DialogActions>
    <Button color="primary">Yes</Button>
```

```
          <Button onClick={closeDelete} color="primary" autoFocus>
           No
          </Button>
        </DialogActions>
      </Dialog>
    </Page>
   </Container>
  );
}
```

Profile

```
import React from "react";
import {
  Container,
  Divider,
  Box,
  Card,
  Grid,
  CardContent,
  Typography,
  Button,
  Link,
} from "@material-ui/core";
import Page from "src/component/Page";
import computer from "../../../images/pc.jpeg";

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 ViewProduct = () => {
  return (
    <>
      <Container maxWidth="xl">
        <Page
          style={{ display: "flex", flexDirection: "column" }}
          title="View product"
        >
          <Box pt={3} mb={8}>
            <Typography variant="h3" style={{ marginBottom: "10px" }}>
              <strong>View product</strong>
            </Typography>
            <Divider />
          </Box>

          <Card>
            <CardContent style={{display:'flex',justifyContent:'center', paddingTop:'50p
x'}}>
              <Grid container md={6} direction="column" spacing={3} style={{padding
Bottom:'50px'}}>
                <Grid item>
                  {" "}
                  <Row field="Category Id :" value="1" />
                </Grid>
                <Grid item>
                  {" "}
                  <Row field="Product Name :" value="laptop" />
                </Grid>
                <Grid item>
                  {" "}
                  <Row field="Metric :" value="10 Unit" />
                </Grid>
                <Grid item>
                  {" "}
                  <Row field="Price :" value="$35.5" />
                </Grid>
                <Grid item>
                  <Row field="Currency Type :" value="Currency" />
                </Grid>
                <Grid item>
                  <Row field="Minimum Value of Purchase :" value="2" />
                </Grid>
                <Grid item>
                  <Row field="Payout Frequency :" value="Quaterly" />
                </Grid>
                <Grid item>
                  <Row field="Product Image :" image={computer} />
                </Grid>
                <Grid item>
                    <Link href='/product'>
                    <Button variant='contained' color='primary' size='large'>Close</Button>
                    </Link>

                </Grid>
              </Grid>
            </CardContent>
          </Card>
        </Page>
      </Container>
    </>
  );
};

export default ViewProduct;
```

Persional Information

```
import {
  Typography,
  Box,
  Paper,
  Card,
  CardContent,
  Button,
  Divider,
  TableCell,
  TableContainer,
  TableHead,
  TableRow,
  TableBody,
  Table,
  InputAdornment,
  makeStyles,
  Grid,
  Link,
  Dialog,
  DialogActions,
  DialogContent,
  DialogContentText,
  FormControl,
  MenuItem,
  Select,
  OutlinedInput,
} from "@material-ui/core";
import SearchIcon from "@material-ui/icons/Search";
import React, { useState } from "react";
import Page from "src/component/Page";
import BlockIcon from "@material-ui/icons/Block";
import DeleteIcon from "@material-ui/icons/Delete";
import EditIcon from "@material-ui/icons/Edit";
import VisibilityIcon from "@material-ui/icons/Visibility";
import {
  MuiPickersUtilsProvider,
  KeyboardDatePicker,
} from "@material-ui/pickers";
import Pagination from '@material-ui/lab/Pagination';

import DateFnsUtils from "@date-io/date-fns";
// import SearchFilter from "../../../component/SearchFilter";
import { usePagination } from "@material-ui/lab/Pagination";
const useStyles = makeStyles((theme) => ({
  btn: {
    width: "100%",
    backgroundColor: "black",
    borderBottomLeftRadius: 5,
    borderBottomRightRadius: 5,
  },
  button: {
    minWidth: "initial",
    padding: "6px",
    marginLeft: "7px",
  },
  table: {
    minWidth: 400,
  },
  ul: {
    listStyle: "none",
    padding: 0,
    margin: 0,
    display: "flex",
  },
  button_style: {
    width: 100,
  },
}));
function createData(
  S_No,
  Date,
  Email,
  Currency_Type,
  Anual_Amount,
  System_fee,
  AmountToBeSend,
  PaymentMethod,
  Actions
) {
  return {
    S_No,
    Date,
    Email,
    Currency_Type,
    Anual_Amount,
    System_fee,
    AmountToBeSend,
    PaymentMethod,
    Actions,
  };
}

const rows = [
  createData(
  1,
    "25/03/2021,5:16 PM",
    "johns@mobiloitte.com",
    "EURO",
    "300EURO",
    "NA",
    "150EURO",
    "Paypal"
  ),
  createData(
    2,
    "25/03/2021,5:16 PM",
    "johns@mobiloitte.com",
    "EURO",
    "300EURO",
    "NA",
    "150EURO",
    "Paypal"
  ),
  createData(
    3,
    "25/03/2021,5:16 PM",
    "johns@mobiloitte.com",
    "EURO",
    "300EURO",
    "NA",
    "150EURO",
    "Paypal"
  ),

  createData(
    4,
    "25/03/2021,5:16 PM",
    "johns@mobiloitte.com",
    "EURO",
    "300EURO",
    "NA",
    "150EURO",
    "Paypal"
  ),

  createData(
    5,
    "25/03/2021,5:16 PM",
    "johns@mobiloitte.com",
    "EURO",
    "300EURO",
    "NA",
    "150EURO",
    "Paypal"
  ),
  createData(
    6,
    "25/03/2021,5:16 PM",
    "johns@mobiloitte.com",
    "EURO",
    "300EURO",
    "NA",
      "150EURO",
      "Paypal"
    ),
  ];
  export default function (props) {
    const classes = useStyles();
    const [modal3IsOpen, set3ModalIsOpen] = useState(false);
    const [modal4IsOpen, set4ModalIsOpen] = useState(false);
    const [isBlock, setBlock] = React.useState(false);

    const openBlock = () => {
      setBlock(true);
    };

    const closeBlock = () => {
      setBlock(false);
    };
    const [isDelete, setDelete] = React.useState(false);

    const openDelete = () => {
      setDelete(true);
    };

    const closeDelete = () => {
      setDelete(false);
    };
    const { items } = usePagination({
      count: 10,
    });
    return (
      <Page title="user-management">
        <Box
          container
          maxWidth="lg"
          style={{ height: 1000, backgroundColor: "white" }}
        >
          <Box
            py={2}
            borderBottom={1}
            style={{ borderColor: "#cccccc", margin: 15 }}
          >
            <Typography
              style={{ fontSize: 18, fontWeight: "bold", color: "#343a40" }}
            >
              REVIEW WITHDRAWL
```

```
     </Typography>
    </Box>
    <Divider />
    <Box pb={2}>
     <SearchFilter
      addProps={{ title: "EXPORT CSV" }}
      pdfProps={{ title: "EXPORT PDF" }}
     />
    </Box>
    <Box>
     <TableContainer component={Paper}>
      <Table className={classes.table} aria-label="customized table">
       <TableHead>
        <TableRow>
         <TableCell
          style={{ color: "white", backgroundColor: "#252d47" }}
         >
          S.No
         </TableCell>
         <TableCell
          style={{ color: "white", backgroundColor: "#252d47", }}
          // align="Left"
         >
          <Typography style={{paddingLeft:"20px"}}>
          Date
          </Typography>
         </TableCell>
         {/* <TableCell style={{ color: "blue" }} align="right">
          Email{" "}
         </TableCell> */}
         <TableCell
          style={{ color: "white", backgroundColor: "#252d47" }}
          align="center"
         >
          Currency Type
         </TableCell>
         <TableCell
          style={{ color: "white", backgroundColor: "#252d47" }}
          align="center"
         >
          Annual Amount
         </TableCell>
         <TableCell
          style={{ color: "white", backgroundColor: "#252d47" }}
          align="center"
          >
            System Fee
          </TableCell>
          <TableCell
            style={{ color: "white", backgroundColor: "#252d47" }}
            align="center"
          >
            Amount To Be Send
          </TableCell>
          <TableCell
            style={{ color: "white", backgroundColor: "#252d47" }}
            align="center"
          >
            Payment Method
          </TableCell>
          <TableCell
            style={{ color: "white", backgroundColor: "#252d47" }}
            align="center"
          >
            Actions
          </TableCell>
        </TableRow>
      </TableHead>
      <TableBody>
        {rows.map((row) => (
          <TableRow key={row.name}>
            <TableCell component="th" scope="row">
              {row.S_No}
            </TableCell>
            <TableCell align="center">{row.Date}</TableCell>
            {/* <TableCell align="center">{row.Email}</TableCell> */}
            <TableCell align="center">{row.Currency_Type}</TableCell>
            <TableCell align="center">{row.Anual_Amount}</TableCell>
            <TableCell align="center">{row.System_fee}</TableCell>
            <TableCell align="center">{row.AmountToBeSend}</TableCell>
            <TableCell align="center">{row.PaymentMethod}</TableCell>
            {/* <Box justifyContent="center" display="flex"> */}
            <TableCell align="center">
              <Grid container>
                <Grid item xs={0}>
                  <Link href="/ViewWithdrawl">
                    <Button
                      variant="contained"
                      color="primary"
                      className={classes.button}
                      >
                        <VisibilityIcon style={{ fontSize: "15px" }} />
                      </Button>
                    </Link>
                  </Grid>
                  <Button
                    variant="contained"
                    color="primary"
                    className={classes.button}
                    onClick={openBlock}
                  >
                    <BlockIcon style={{ fontSize: "15px" }} />
                  </Button>
                  <Button
                    variant="contained"
                    color="secondary"
                    className={classes.button}
                    onClick={openDelete}
                  >
                    <DeleteIcon style={{ fontSize: "15px" }} />
                  </Button>
                </Grid>
              </TableCell>
              {/* </Box> */}
            </TableRow>
          ))}
        </TableBody>
      </Table>

    </TableContainer>
    <Box display="flex" justifyContent="flex-end">
    <Pagination count={10} shape="rounded" />
    </Box>
  </Box>
</Box>
<Dialog
  open={isBlock}
  onClose={closeBlock}
  aria-labelledby="alert-dialog-title"
  aria-describedby="alert-dialog-description"
>
  <DialogContent>
    <DialogContentText id="alert-dialog-description">
    Are you sure you want to block this Review?
    </DialogContentText>
```

```
      </DialogContent>
      <DialogActions>
        <Button color="primary">Yes</Button>
        <Button onClick={closeBlock} color="primary" autoFocus>
          No
        </Button>
      </DialogActions>
    </Dialog>
    <Dialog
      open={isDelete}
      onClose={closeDelete}
      aria-labelledby="alert-dialog-title"
      aria-describedby="alert-dialog-description"
    >
      <DialogContent>
        <DialogContentText id="alert-dialog-description">
        Are you sure you want to delete this Review?
        </DialogContentText>
      </DialogContent>
      <DialogActions>
        <Button color="primary">Yes</Button>
        <Button onClick={closeDelete} color="primary" autoFocus>
          No
        </Button>
      </DialogActions>
    </Dialog>
  </Page>
 );
}

const SearchFilter = ({ exportProps, addProps, pdfProps }) => {
 const [selectedDate, setSelectedDate] = React.useState(
   new Date("2014-08-18T21:11:54")
 );

 const handleDateChange = (date) => {
   setSelectedDate(date);
 };

 return (
   <Card raised>
     <CardContent>
       <Typography variant="h4" style={{ marginBottom: "8px" }}>
         Filter by
       </Typography>
```

```
<Grid container spacing={1} alignItems="center">
  <Grid item md={2}>
    <Box style={{ marginBottom: 5 }}>
      <Typography style={{ paddingBottom: 15 }} variant="h5">
        Status
      </Typography>

      <Grid item md={12}>
        <FormControl variant="outlined" fullWidth>
          <Select
            // labelId="demo-simple-select-outlined-label"
            id="demo-simple-select-outlined"
          >
            <MenuItem value={10}>All</MenuItem>
            <MenuItem value={20}>Paypal</MenuItem>
            <MenuItem value={30}>Bank</MenuItem>
          </Select>
        </FormControl>
      </Grid>
    </Box>
  </Grid>

  <Grid item md={2}>
    <Box>
      <Typography variant="h4">From Date</Typography>
      <MuiPickersUtilsProvider utils={DateFnsUtils}>
        <KeyboardDatePicker
          inputVariant="outlined"
          disableToolbar
          variant="inline"
          format="MM/dd/yyyy"
          margin="normal"
          id="date-picker-inline"
          value={selectedDate}
          onChange={handleDateChange}
          KeyboardButtonProps={{
            "aria-label": "change date",
          }}
        />
      </MuiPickersUtilsProvider>
    </Box>
  </Grid>
  <Grid item md={2}>
    <Box>
```

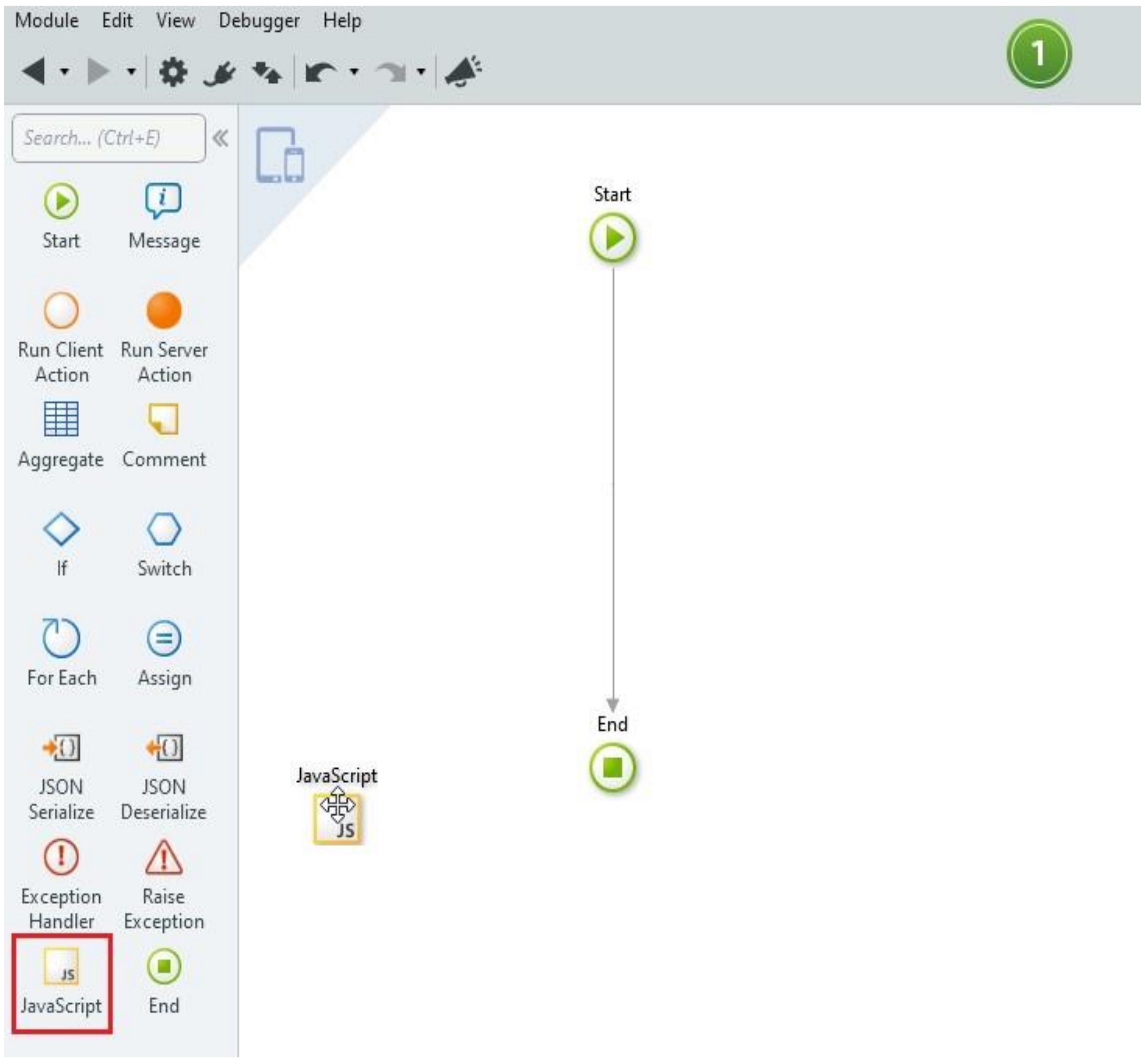
```
            <Typography variant="h4">To Date</Typography>
            <MuiPickersUtilsProvider utils={DateFnsUtils}>
              <KeyboardDatePicker
                inputVariant="outlined"
                disableToolbar
                variant="inline"
                format="MM/dd/yyyy"
                margin="normal"
                id="date-picker-inline"
                value={selectedDate}
                onChange={handleDateChange}
                KeyboardButtonProps={{
                  "aria-label": "change date",
                }}
              />
            </MuiPickersUtilsProvider>
          </Box>
        </Grid>
        <Grid item md={2}>
          <Box style={{ marginTop: 30,width:210 }}>
            <FormControl variant="outlined">
              <OutlinedInput
                placeholder="search by name"
                endAdornment={
                  <InputAdornment position="end">
                    <SearchIcon
                      aria-label="toggle password visibility"
                      edge="end"
                    ></SearchIcon>
                  </InputAdornment>
                }
              />
            </FormControl>
          </Box>
        </Grid>
        <Grid item>
          <Box style={{ paddingTop: 48,marginLeft:10 }}>
            <Button variant="contained" color="primary">
              Search
            </Button>
            <Button
              variant="contained"
              color="primary"
              style={{ marginLeft: "3px", marginRight: "3px" }}
            >
            Reset
          </Button>
        </Box>
        {exportProps && (
          <Button
            variant="contained"
            color="primary"
            onClick={exportProps.onClick}
          >
            {exportProps.title}
          </Button>
        )}
      </Grid>
    </Grid>

    <Box display="flex" justifyContent="flex-end" py={4}>
      {addProps && (
        <Button
          style={{ marginRight: 10 }}
          variant="contained"
          color="primary"
        >
          {addProps.title}
        </Button>
      )}
      {pdfProps && (
        <Button variant="contained" color="primary">
          {pdfProps.title}
        </Button>
      )}
    </Box>
   </CardContent>
  </Card>
 );
};
```

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.

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 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.
- 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.

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

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.

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

1. Rodriguez, S., E. N. Baydak, F. F. Schoeggl, S. D. Taylor, G. Hay, and H. W. Yarranton. "Regular solution based approach to modeling asphaltene precipitation from native and reacted oils: Part 3, visbroken oils." *Fuel* 257 (2019): 116079.

2. Tabarés, Raúl. "HTML5 and the evolution of HTML; tracing the origins ofdigital platforms." *Technology in Society* 65 (2021): 101529.

3. Chen, Peihong, Jinling Liu, Kaijun Zhang, Dongzhen Huang, Siyu Huang, Qingchun Xie, Fan Yang et al. "Preparation of clarithromycin floating core-shell systems (CSS) using multi-nozzle semi-solid extrusion-based 3D printing." *International Journal of Pharmaceutics* (2021): 120837.

4. Mojirsheibani, Majid. "A note on the performance of bootstrap kernel density estimation with small re-sample sizes." *Statistics & Probability Letters* (2021): 109189.

5. Alex, Scaria, and T. Dhiliphan Rajkumar. "2 Spider Bird Swarm Algorithm with Deep Belief Network for Malicious JavaScript Detection." *Computers &Security* (2021): 102301.

6. Hallensleben, Cynthia, Eline Meijer, Jaco Biewenga, Regien MM Kievits- Smeets, Marjan Veltman, Xiaoyue Song, Job FM van Boven, and Niels H. Chavannes. "REducing Delay through edUcation on eXacerbations (REDUX)in patients with COPD: a pilot study." *Clinical eHealth* 3 (2020): 63-68.

7. Hlina, Benjamin L., Daniel M. Glassman, Auston D. Chhor, Brooke S. Etherington, Chris K. Elvidge, Benjamin K. Diggles, and Steven J. Cooke."Hook retention but not hooking injury is associated with behavioral differences in Bluegill." *Fisheries Research* 242 (2021): 106034.

8. Khanna, Reena, Brian Bressler, Barrett G. Levesque, Guangyong Zou, Larry W. Stitt, Gordon R. Greenberg, Remo Panaccione et al. "Early combined immunosuppression for the management of Crohn's disease (REACT): a cluster randomised controlled trial." *The Lancet* 386, no. 10006 (2015): 1825-1834

9. David, Matthew. *Developing websites with jQuery mobile*. Focal Press, 2011..

10. Hayashi, Eriko. "JS6-3 Cancer treatment support for patients who have difficulties in daily life." *Annals of Oncology* 32 (2021): S238.

11. Imamura, Yoshinori. "JS5-3 Current status and core measures of cancer-associated thromboembolism." *Annals of Oncology* 32 (2021): S237.

12. Gao, Ya, Ziyu Zhu, Xiaoxue Xi, Tingwei Cao, Wei Wen, Xiuhua Zhang, and Shengfu Wang. "An aptamer-based hook-effect-recognizable three-line lateral flow biosensor for rapid detection of thrombin." *Biosensors and*

Bioelectronics 133 (2019): 177-182.

13. Merz, Justin, Varaprasad Bandaru, Quinn Hart, Nathan Parker, and Bryan M. Jenkins. "Hybrid Poplar based Biorefinery Siting Web Application (HP-BiSWA): An online decision support application for siting hybrid poplar based biorefineries." *Computers and Electronics in Agriculture* 155 (2018): 76-83.

14. Olejnik, Richard, Teodor-Florin Fortiş, and Bernard Toursel. "Webservices oriented data mining in knowledge architecture." *Future Generation Computer Systems* 25, no. 4 (2009): 436-443

15. Douglas. *JavaScript: : The Good Parts*. " O'Reilly Media, Inc.", 2008.

,

.

.