Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CSE 3002- INTERNET AND WEB PROGRAMMING

Fall Semester 2022-23

Slot:L27+L28

ONLINE CURRENCY TRADING WITH DIGITAL WALLET

SUBMITTED BY:

ADITYA KUMAR (20BCE0428)

GAURAV GATTANI (20BCE2250)

PRASISH BHATTA (20BCE2809)

Declaration:

I hereby declare that the thesis entitled “ONLINE CURRENCY TRADING USING DIGITAL WALLET” submitted by us, for the award of the degree of Bachelor of Technology in CSE CORE, CSE With Specialization in Information Security to VIT is a record of bonafide work carried out by me under the supervision of Mythili T.
We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other Institute or university.

Place: VIT, Vellore

Date: 3/11/2022

ABSTRACT

In electronic commerce, the challenges of the payment system were initially underestimated. Doing business via the internet and mobile telephony has so far been dominated by payment methods in traditional companies. However, in light of advances in e-commerce, traditional economic models are increasingly reaching their limits. To understand the concept of e-commerce, E-wallet is a convenient, easy-to-use and secure global payment system. It is a flexible "personal banking system" with a number of payment and payment options.

Physical wallets present numerous problems: when you're at a cash register, getting the right amount of cash or the right combination of payment, discount, and loyalty cards can take a long time. Worse still, if your wallet is lost or stolen, you need to remember which cards you had and manually cancel each one, as well as apply for new ID cards or documents: driver's license, health card, car insurance card, and soon. One solution to these problems is to replace the physical wallet with a digital wallet built into an existing mobile device, such as a cell phone. This would allow users to manage multiple monetary and identification instruments and quickly search for them by name, type, or other keywords. Additionally, a digital wallet would increase security as all data would be encrypted and backup options would make it easier to recover from loss or theft.

SURVEY

Traditionally, online wallets do not guarantee customers that they are safe with the type of wallet they choose. It has been proven over time that there is no technology that can strengthen the banking system and protect the rights of customers and clients against financial fraud. Since goodwill comes at a cost, it is a trust that takes years to build. The problem with wallets is that they have a certain

Advancement of technology has paved the way for a digitized world. Everything is getting digitized nowadays including the various aspects of money for e.g., currency and transfer of money. The main theme of our project is currency trading system that allows trading using coins with an added digital wallet feature. Our main motivation for the project is the rise in the use of the various digital currencies and its rapid growth in use and value.

Challenges of the existing systems:

1. The existing websites doesn't contain both coin and wallet system.

2. Extra payment is charged when converted to other currencies.

3. Many customers still believe that there is no additional value in using e-wallets.

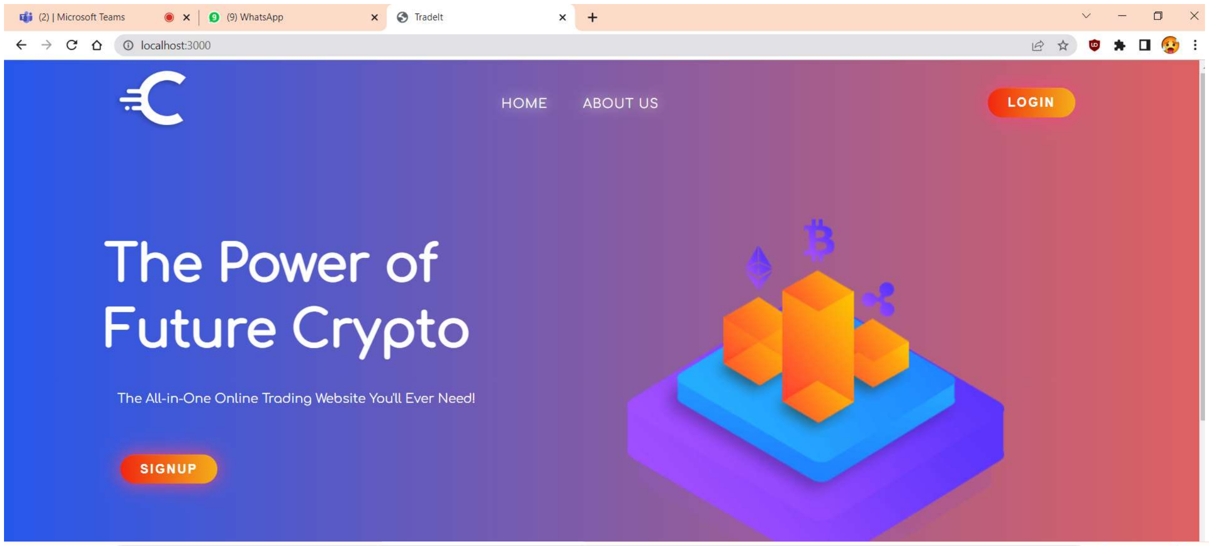
4. People don't trust these kinds of applications due to less security.

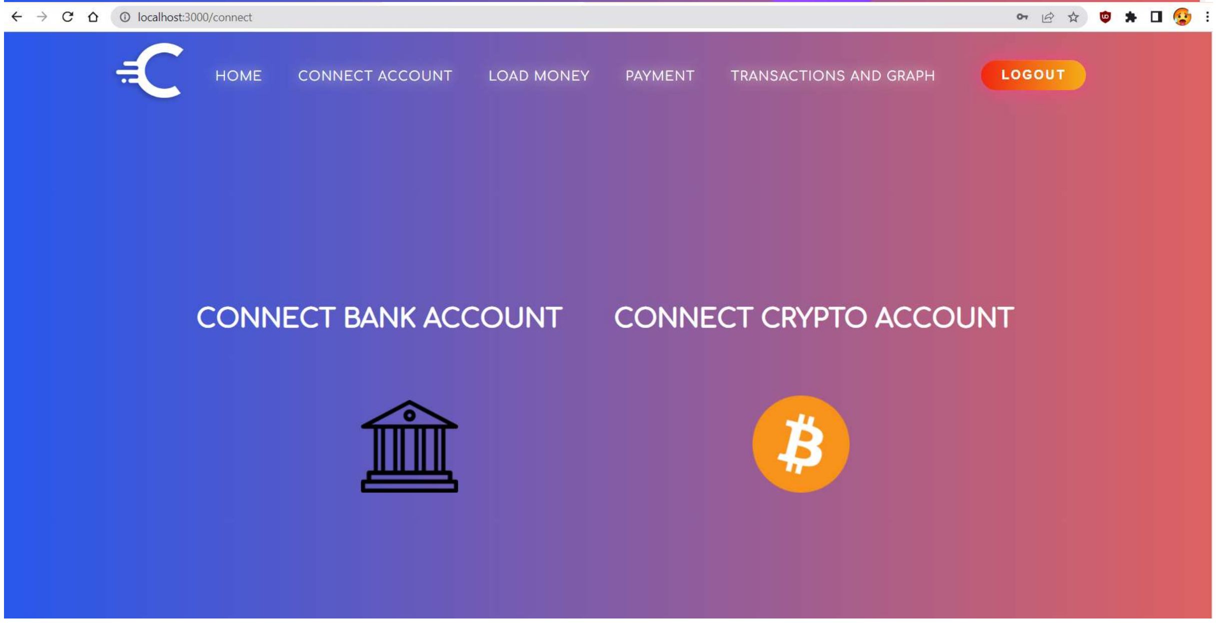
Motivation:

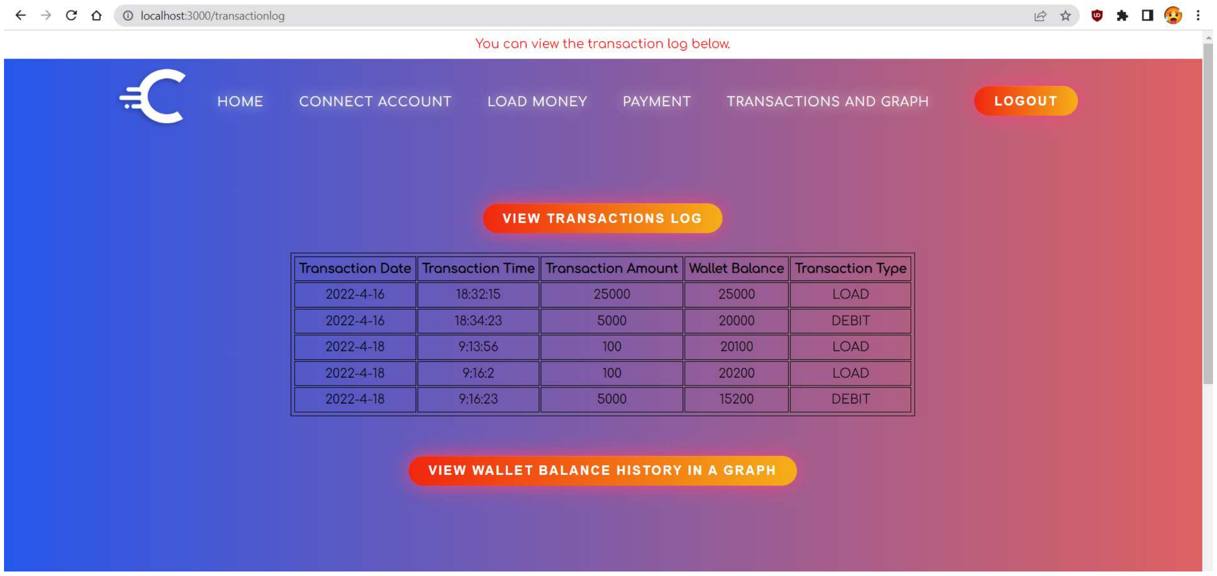
The existing system is focused only on either of digital wallet or cryptocurrency trading. That’s why our project is aimed to develop a system which consists of feature for both coin and wallet transaction. Here we have included the option for payment using the digital wallet and cryptocurrency. The digital wallet can be loaded through bank account and the cryptocurrency can be loaded through crypto account. Hence making it easier for the small to huge number of transactions. And one of the chief factors which motivated us for doing this project is the covid-19 pandemic. Because of the fear of the COVID, people used to feel afraid for the trading of the cash. Hence, we came up with the idea of creating a website for the digital trading of the currency via our website and the feature of the digital wallet present there.

OBJECTIVES

SAMPLES:

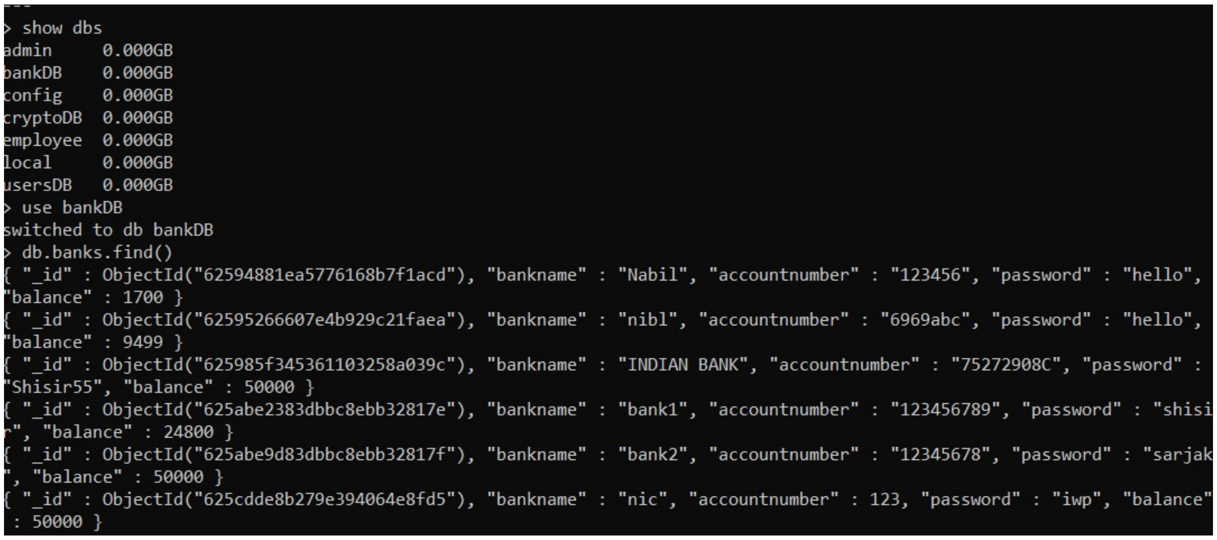

(2) | Microsoft Teams
(9) WhatsApp
TradeIt
localhost:3000
HOME
ABOUT US
LOGIN
The Power of Future Crypto
The All-in-One Online Trading Website You'll Ever Need!
SIGNUP





You can view the transaction log below.

Transaction Date	Transaction Time	Transaction Amount	Wallet Balance	Transaction Type
2022-4-16	18:32:15	25000	25000	LOAD
2022-4-16	18:34:23	5000	20000	DEBIT
2022-4-18	9:13:56	100	20100	LOAD
2022-4-18	9:16:2	100	20200	LOAD
2022-4-18	9:16:23	5000	15200	DEBIT



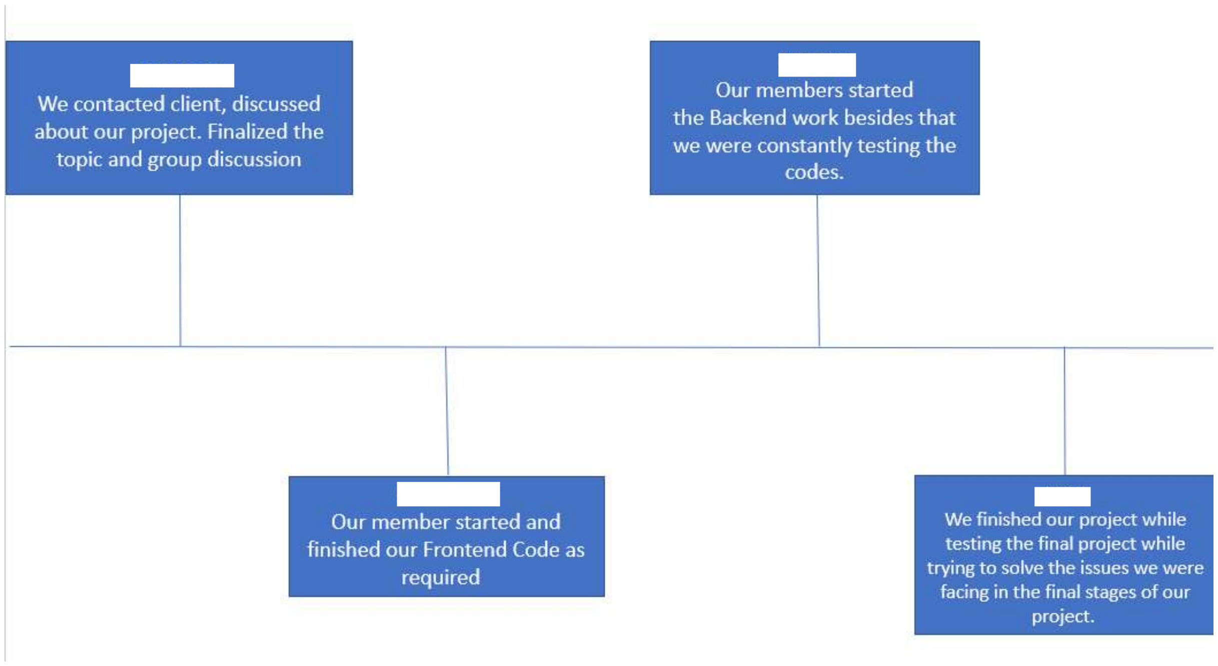
```
> show dbs
admin     0.000GB
bankDB    0.000GB
config    0.000GB
cryptoDB  0.000GB
employee  0.000GB
local     0.000GB
usersDB   0.000GB
> use bankDB
switched to db bankDB
> db.banks.find()
{ "_id" : ObjectId("62594881ea5776168b7f1acd"), "bankname" : "Nabil", "accountnumber" : "123456", "password" : "hello",
"balance" : 1700 }
{ "_id" : ObjectId("62595266607e4b929c21faea"), "bankname" : "nibl", "accountnumber" : "6969abc", "password" : "hello",
"balance" : 9499 }
{ "_id" : ObjectId("625985f345361103258a039c"), "bankname" : "INDIAN BANK", "accountnumber" : "75272908C", "password" :
"Shisir55", "balance" : 50000 }
{ "_id" : ObjectId("625abe2383dbbc8ebb32817e"), "bankname" : "bank1", "accountnumber" : "123456789", "password" : "shisi
r", "balance" : 24800 }
{ "_id" : ObjectId("625abe9d83dbbc8ebb32817f"), "bankname" : "bank2", "accountnumber" : "12345678", "password" : "sarjak
", "balance" : 50000 }
{ "_id" : ObjectId("625cdde8b279e394064e8fd5"), "bankname" : "nic", "accountnumber" : 123, "password" : "iwp", "balance"
 : 50000 }
```

- To Design a website for trading money digitally without any physical movement.

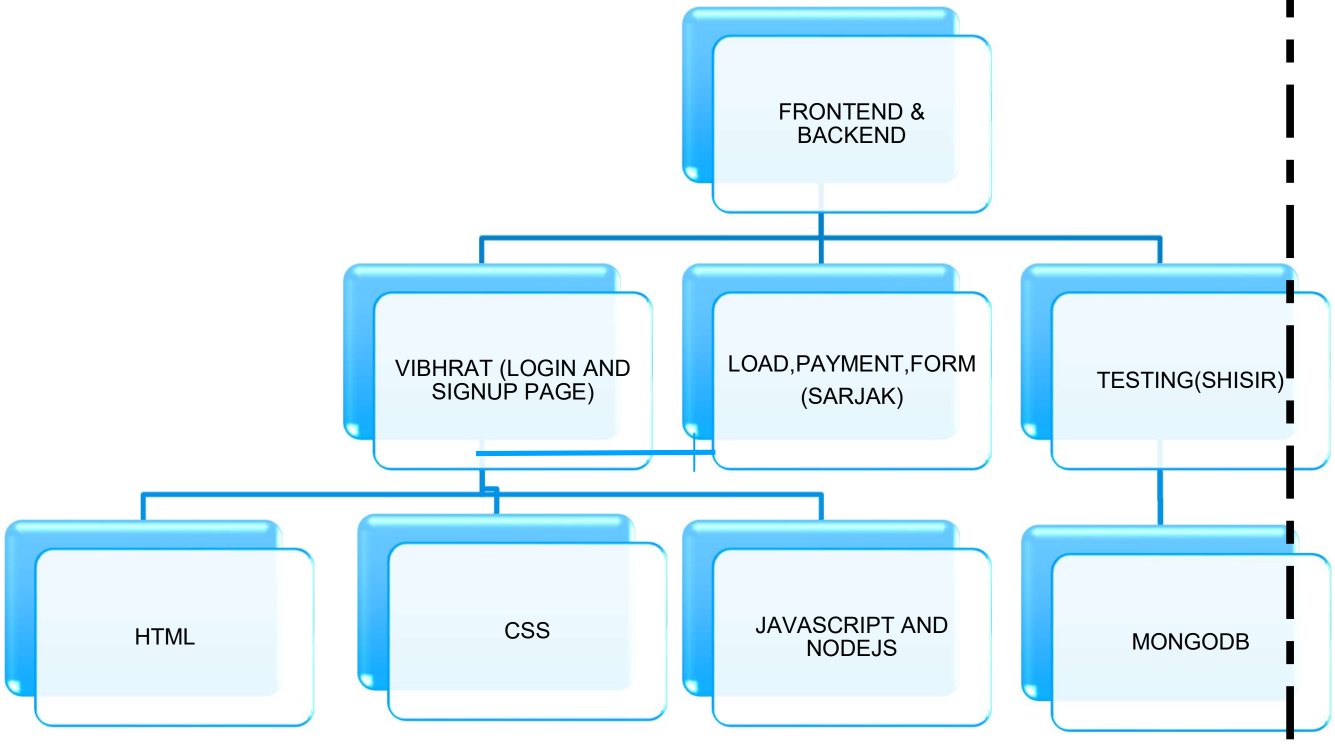
- To design an e-wallet where the money could be loaded from the bank account or the crypto account.

- Using mongo DB for the database storage which includes storing the User information, Transaction history, bank and crypto accounts credentials and the data storing.

ARCHITECTURE DIAGRAM:


We contacted client, discussed about our project. Finalized the topic and group discussion
Our members started the Backend work besides that we were constantly testing the codes.
Our member started and finished our Frontend Code as required
We finished our project while testing the final project while trying to solve the issues we were facing in the final stages of our project.

WORK CHART:

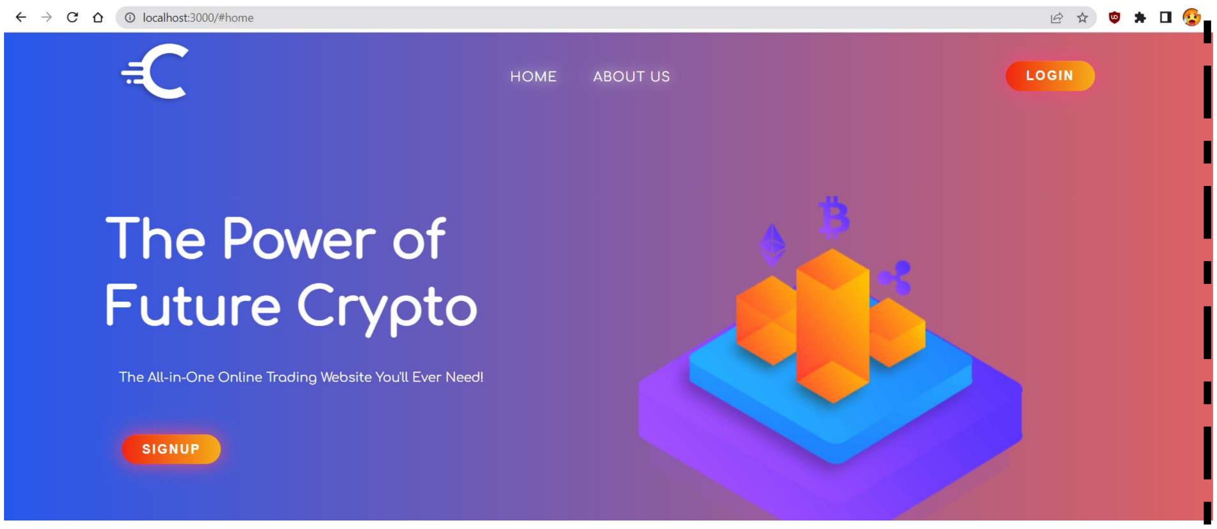


MODULE WISE DESCRIPTION:

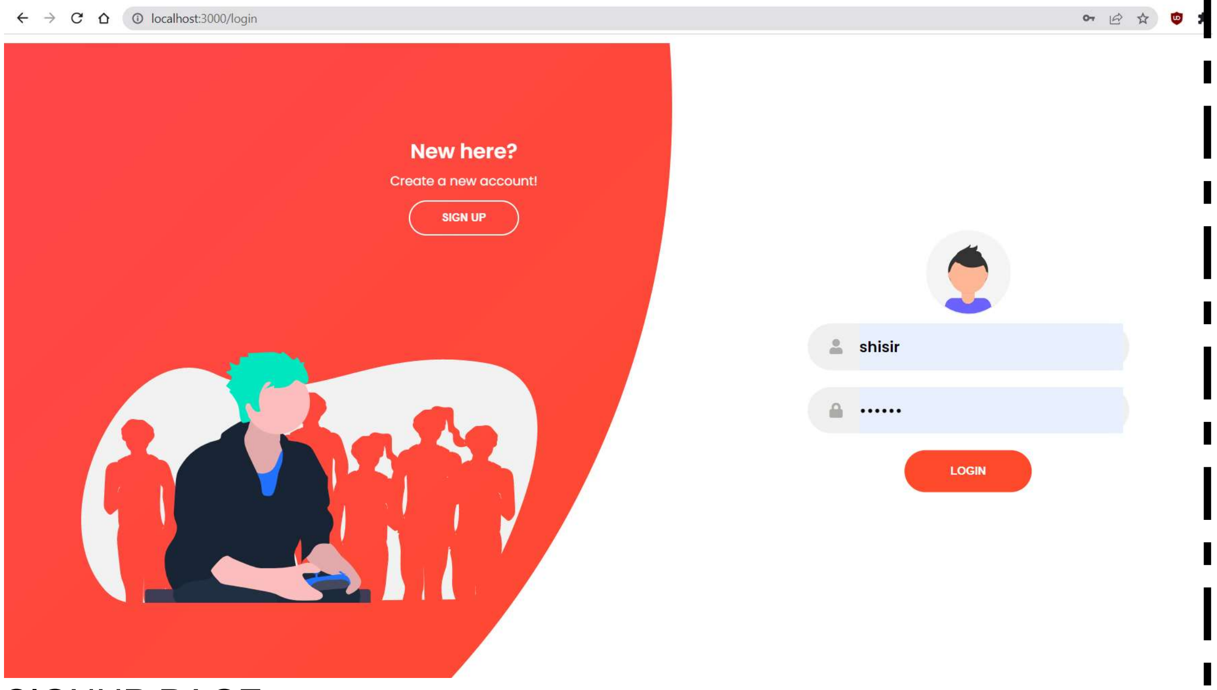
HTML & CSS FOR ALL THE PAGES:

On the month of May, we were able to write html and css code for following pages:

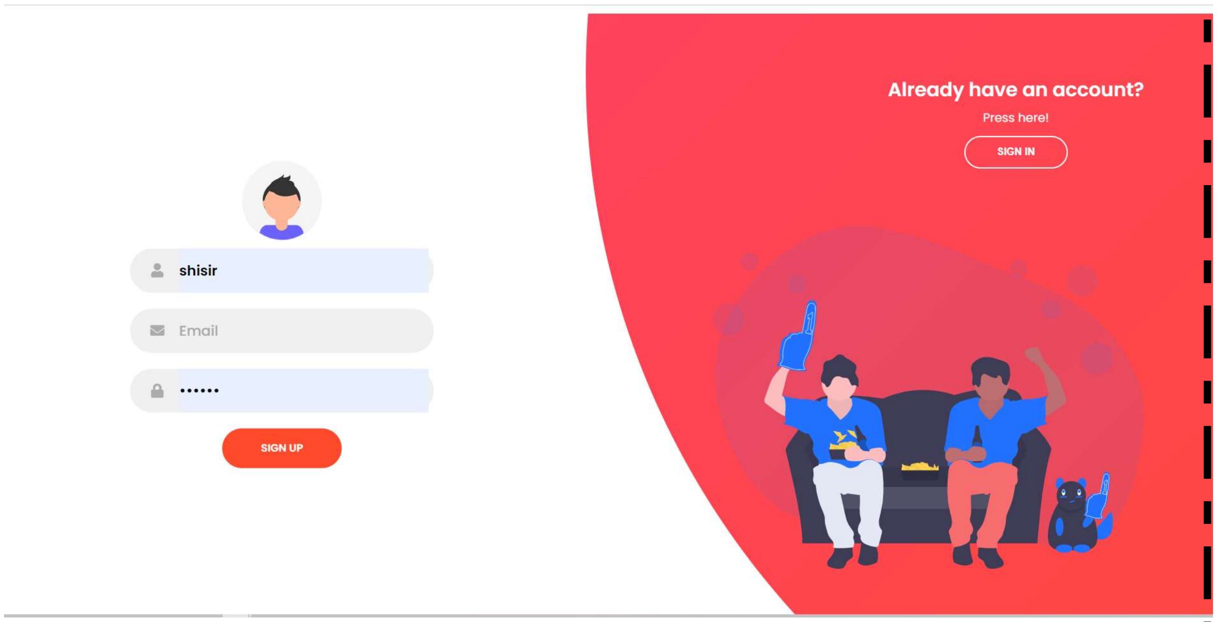
- HOME PAGE


localhost:3000/#home
HOME
ABOUT US
LOGIN
The Power of Future Crypto
The All-in-One Online Trading Website You'll Ever Need!
SIGNUP

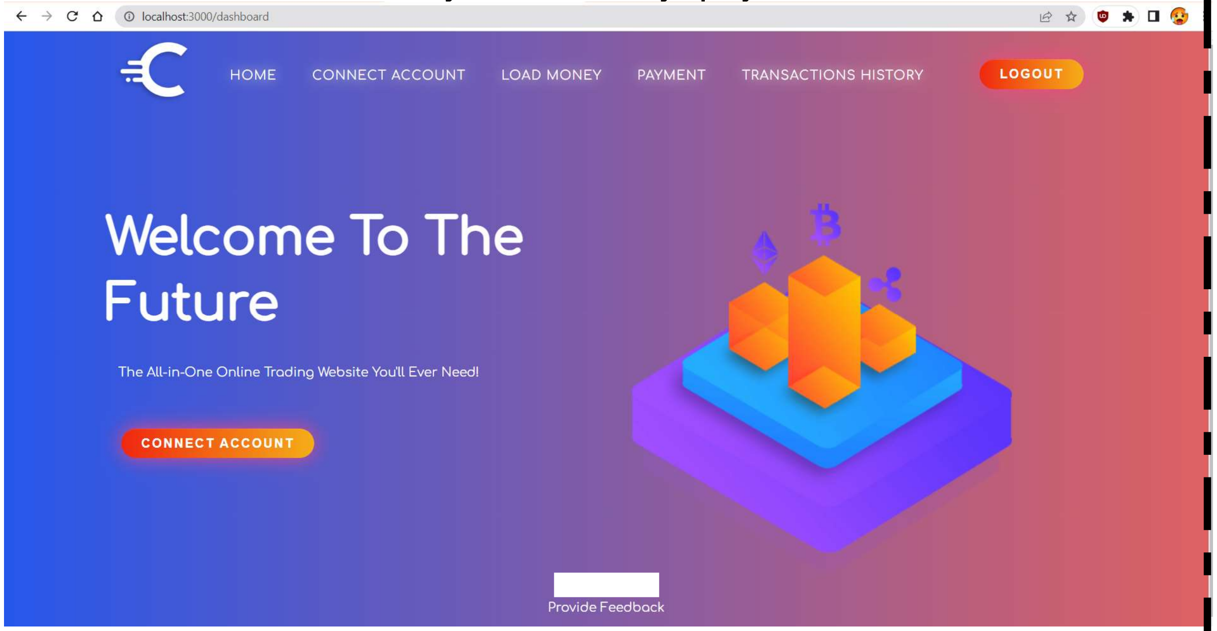
- LOGIN PAGE



- SIGNUP PAGE

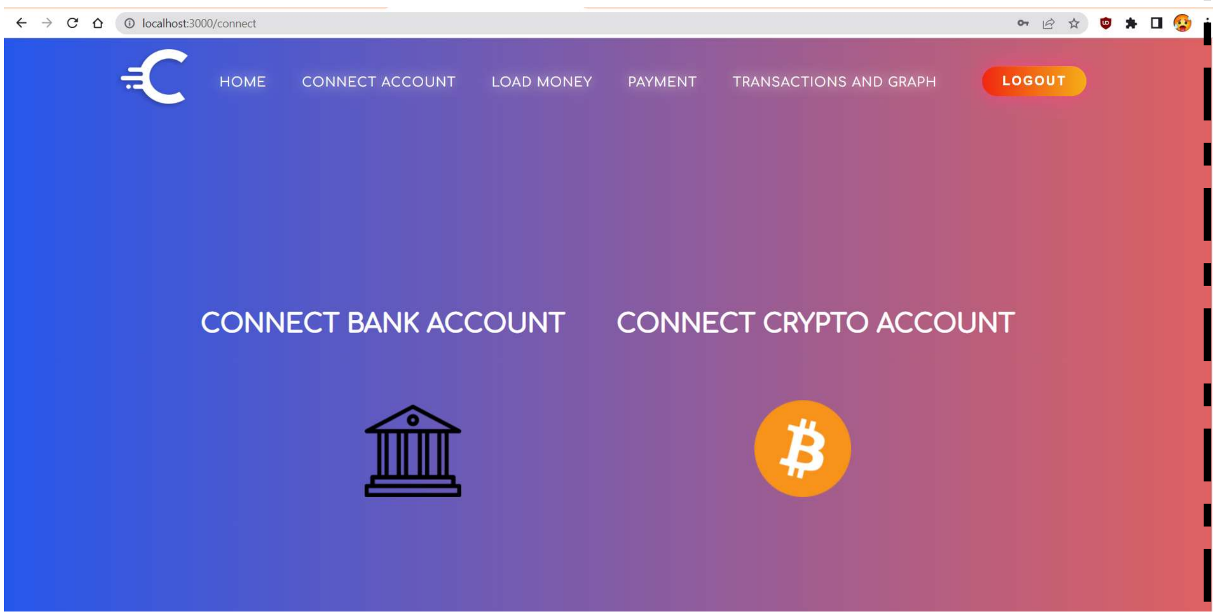


- DASHBOARD PAGE-
  Here, in this page we can see the features like: connect account, and in the navigation bar we can see different menus like connect account, transaction history, load money, payment.

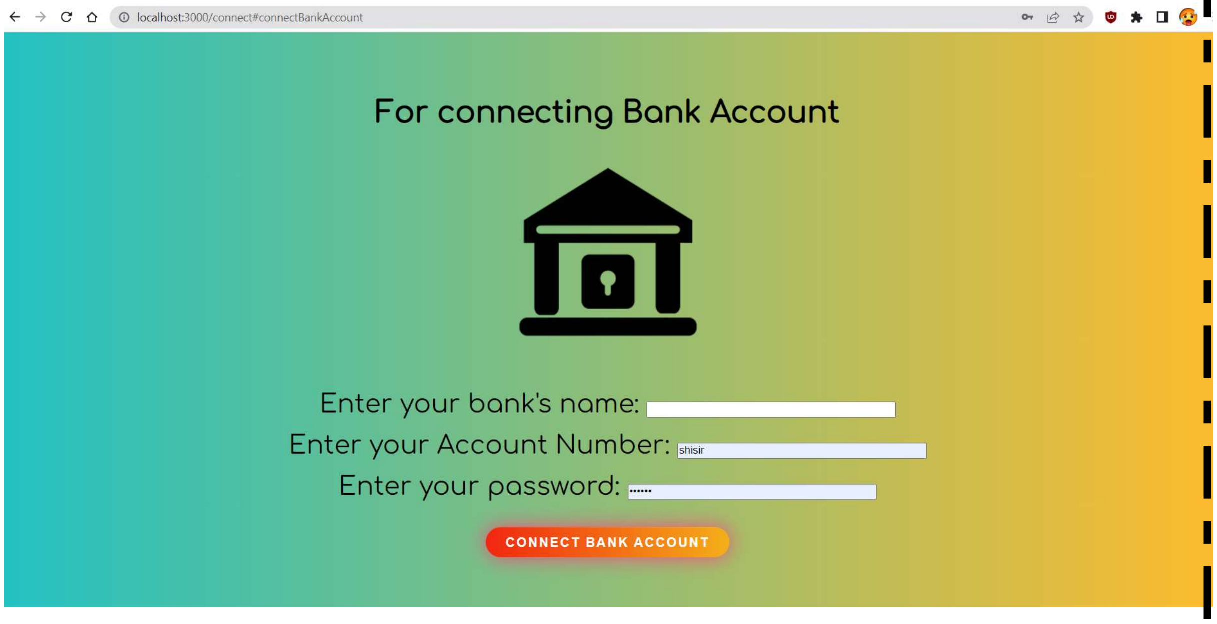


- CONNECT ACCOUNT PAGE:

In this page we can see the options to connect the either of bank or a crypto account.



- LINKING BANK ACCOUNT PAGE:



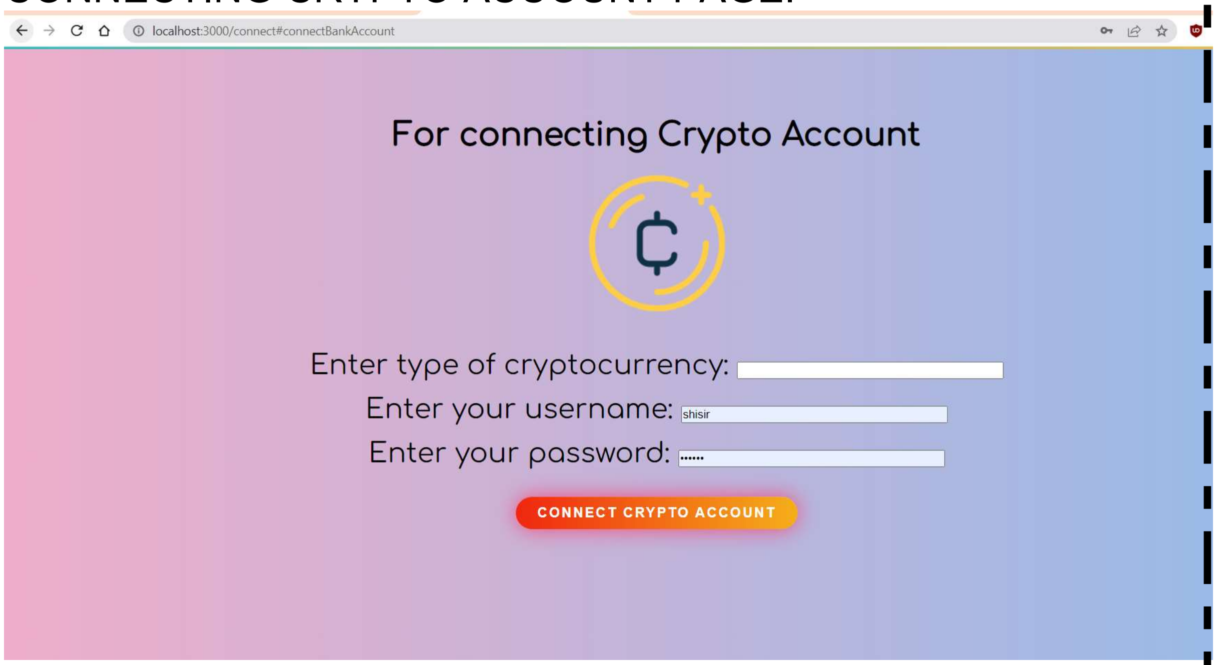
For connecting Bank Account

Enter your bank's name:

Enter your Account Number: shisir

Enter your password: ••••••

- CONNECTING CRYPTO ACCOUNT PAGE:



localhost:3000/connect#connectBankAccount

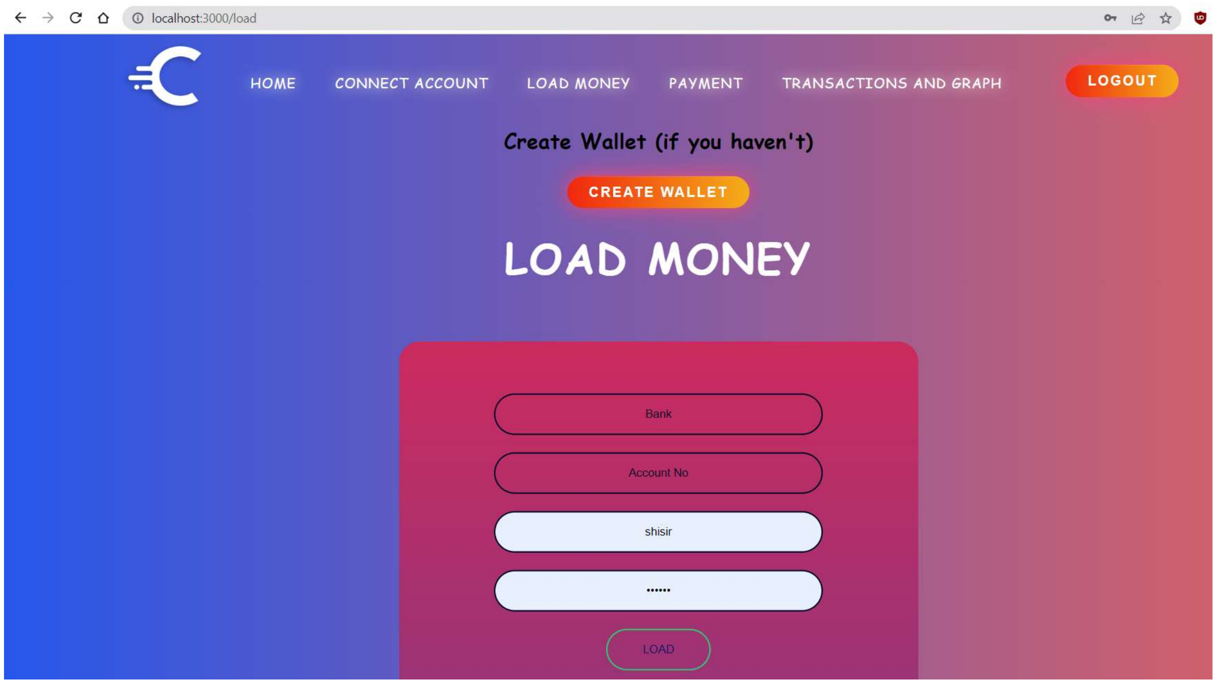
For connecting Crypto Account

Enter type of cryptocurrency:

Enter your username: shisir

Enter your password: ••••••

- LOAD MONEY PAGE



localhost:3000/load

Create Wallet (if you haven't)

LOAD MONEY

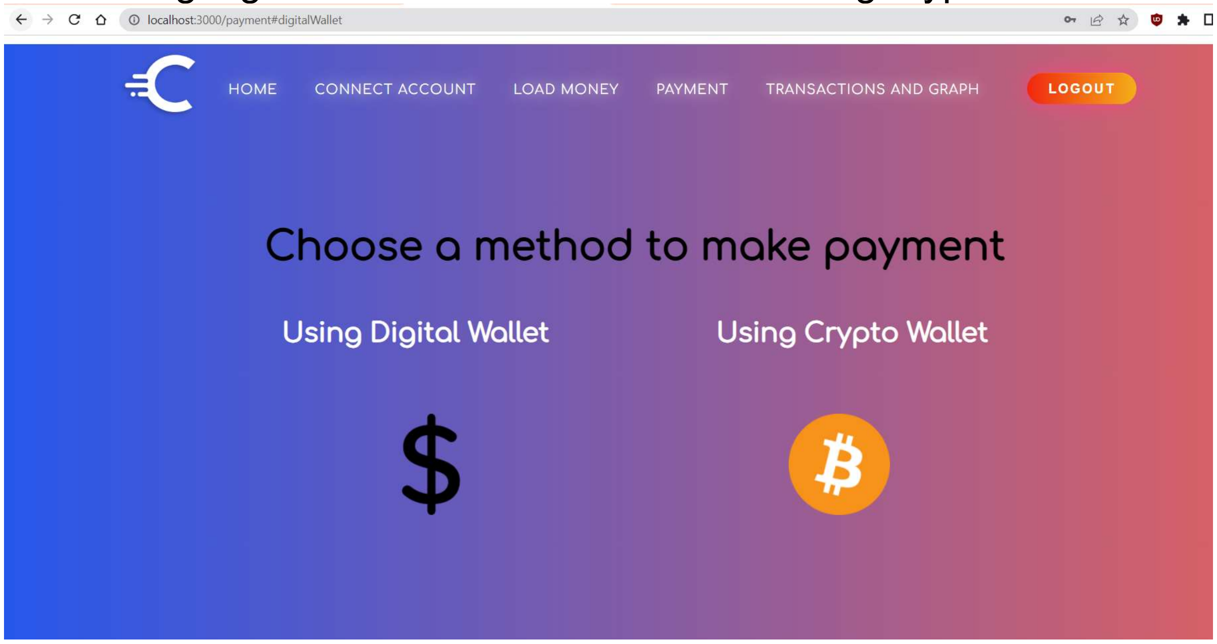
Bank

Account No

shisir

••••••

- PAYMENT METHOD PAGE:
  Here in this page we can see two options: making payment Using digital wallet and another one is using crypto wallet.

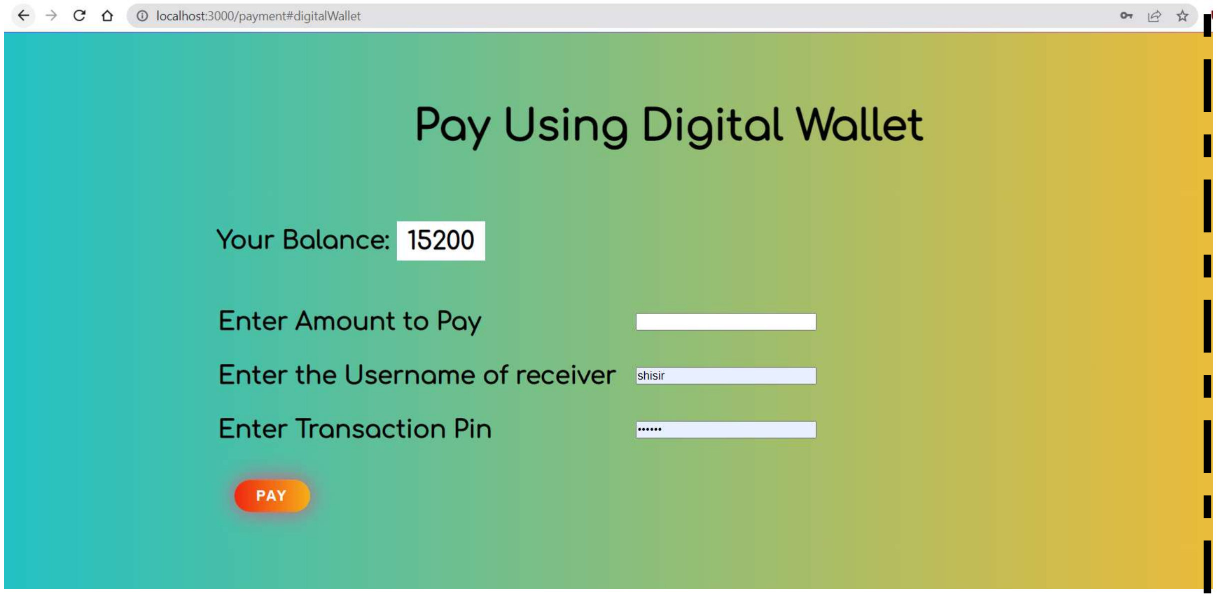


Choose a method to make payment

Using Digital Wallet

Using Crypto Wallet

- Digital wallet page:
  Whenever clicked on digital wallet then the following will appear:



Pay Using Digital Wallet

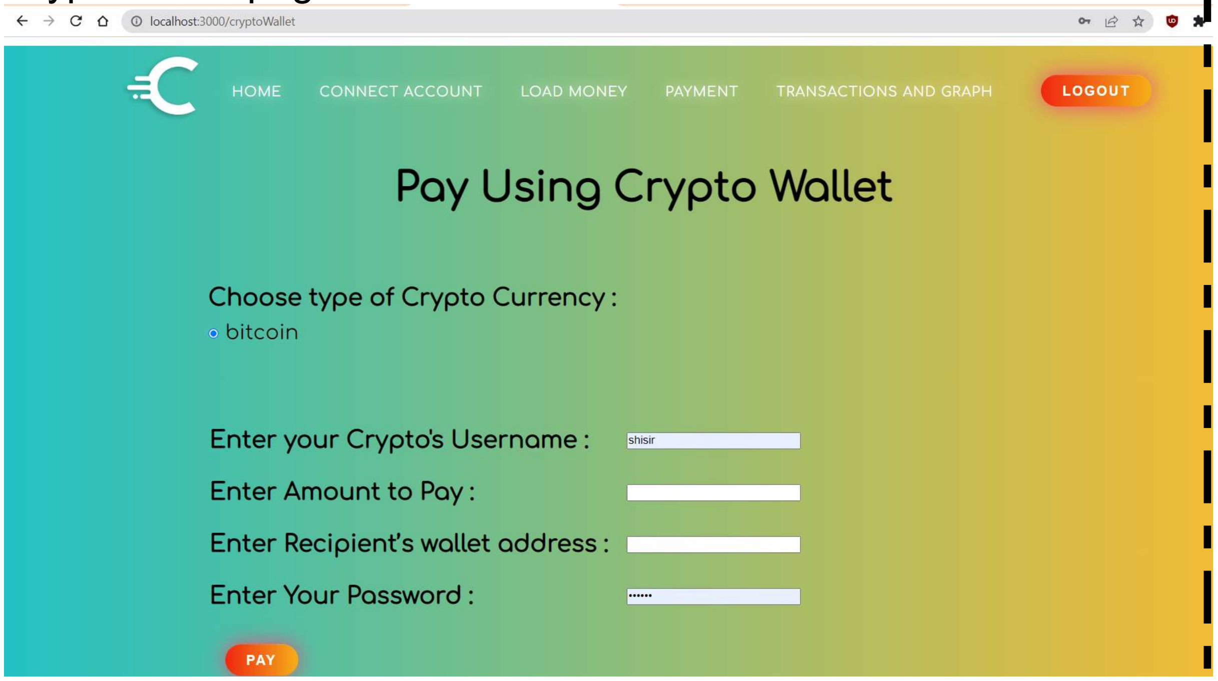
Your Balance: 15200

Enter Amount to Pay

Enter the Username of receiver shisir

Enter Transaction Pin ••••••

- Crypto wallet page:

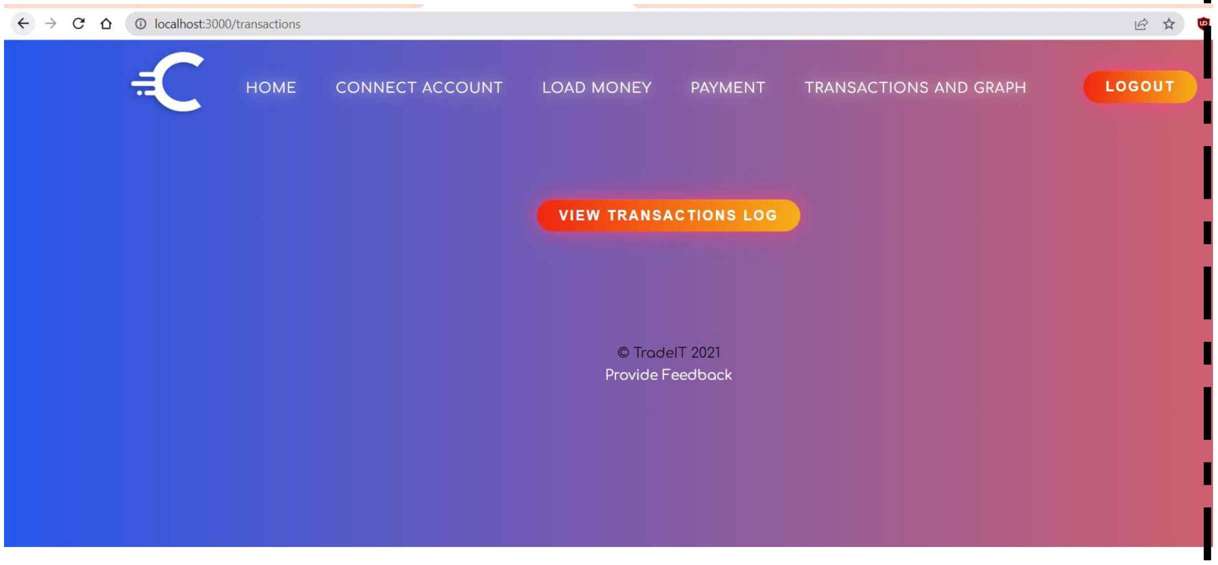


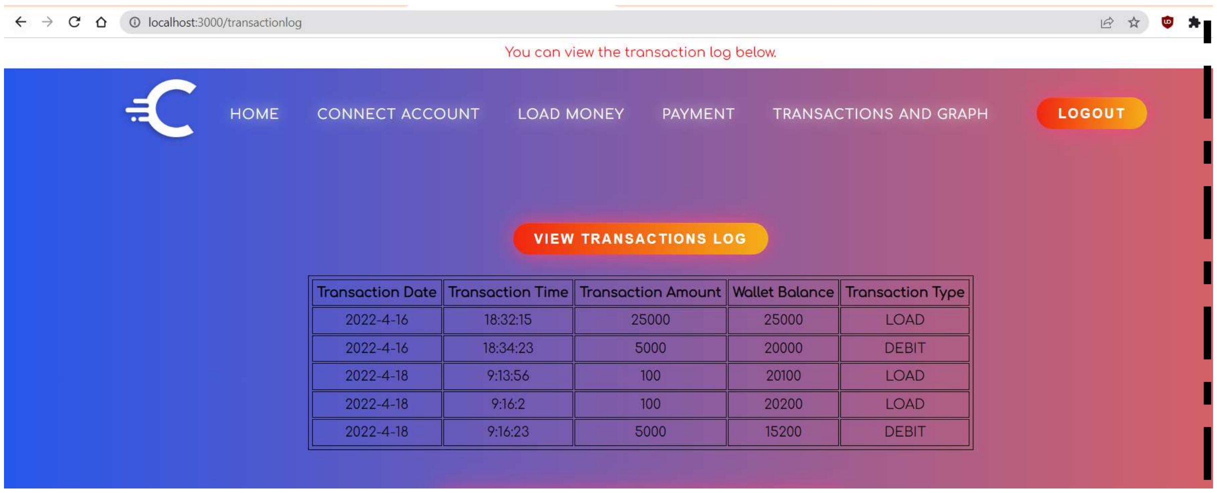
Pay Using Crypto Wallet

Choose type of Crypto Currency :
● bitcoin

Enter your Crypto's Username : shisir
Enter Amount to Pay :
Enter Recipient's wallet address :
Enter Your Password : ••••••

- Transaction history page:
  In this page we can see a button for viewing the transaction history in a log. When clicked on that button following will be shown





You can view the transaction log below.

Transaction Date	Transaction Time	Transaction Amount	Wallet Balance	Transaction Type
2022-4-16	18:32:15	25000	25000	LOAD
2022-4-16	18:34:23	5000	20000	DEBIT
2022-4-18	9:13:56	100	20100	LOAD
2022-4-18	9:16:2	100	20200	LOAD
2022-4-18	9:16:23	5000	15200	DEBIT

So these are all the pages that were built using html css and java script.

AFTER COMPLETING ALL THE FRONTEND DESIGNING TASK, ON THE MONTH OF MARCH WE STARTED AND FINISHED OUR BACKEND TASKS.

WE USED NODEJS AND MONGODB HERE
MONGO DB WAS USED IN ORDER TO STORE ALL THE NECESSARY DATAS INTO DIFFERENT DATABASES.

BACKEND CONFIGURATION USING JS AND USING MONGO DB SCHEMA FOR IT. THE CODE IS SHOWN BELOW:

```
const express = require("express");

const bodyParser = require("body-parser");

const mongoose = require("mongoose");

var Float = require('mongoose-float').loadType(mongoose);

const GoogleChartsNode = require('google-charts-node');



const app = express();
```

```
app.set('view engine','ejs');

app.use(express.static("public"));

app.use(bodyParser.urlencoded({
  extended: true
}));


let currentUser;

// DATABASE CONNECTION

// mongoose.connect("mongodb://localhost: 27017/usersDB", {
//   useNewUrlParser: true
// }, {
//   useUnifiedTopology: true
// }).catch(error => handleError(error));;



const cryptoSchema = new mongoose.Schema({
  cryptoname: {
    type: String,
    // required: [true, "Please enter crypto's name"]
  },
  username: {
    type: String,
    // required: true
  },
  password: {
    type: String,
    // required: true
  },
  address: String,
  balance: {
    type: Float
  }
});

const bankSchema = new mongoose.Schema({

  bankname: {
    type: String,
    // required: [true, "Please enter bank's name"]
  },
  accountnumber: {
    type: String,
    // required: true
  },
  password: {
    type: String,
    // required: true
  },
  balance: {
    type: Float
  }
});

const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: [true, 'Please enter username']
  },
  email: {
    type: String,
    lowercase: true
  },
  password: {
    type: String,
    required: true
  }

});

const connectBankSchema = new mongoose.Schema({
  username: String,
  bankname: {
    type: String,
    // required: [true, "Please enter bank's name"]
  },
  accountnumber: {
    type: String,
    // required: true
  },
  password: {
    type: String,
    // required: true
  },
  balance: {
    type: Float
  },
  pin: Number
});
```

```
const connectCryptoSchema = new mongoose.Schema({
  user: String,
  cryptoname: {
    type: String,
    // required: [true, "Please enter crypto's name"]
  },
  username: {
    type: String,
    // required: true
  },
  password: {
    type: String,
    // required: true
  },
  address: String,
  balance: {
    type: Float
  }
});

const walletSchema = new mongoose.Schema({
  user: String,
  balance: {
    type: Float
  },
  transactionPin: Number
});

const transactionSchema = new mongoose.Schema({
  user: String,
  transactionamt: Float,

  balance: Float,

  transactiontype: String,

  date: String,

  time: String

});


const feedbackSchema = new mongoose.Schema({

  user: String,

  rating: String,

  feedback: String

});










app.get("/", function(req, res) {

  res.sendFile(__dirname + "/home.html");

});


app.get("/login", function(req, res) {

  res.sendFile(__dirname + "/form.html");

});


app.get("/dashboard", function(req, res) {

  res.sendFile(__dirname + "/dashboard.html");

});
```

```
app.get("/connect", function(req, res) {
  res.sendFile(__dirname + "/connect.html");
});


app.get("/cryptoWallet",function(req,res){


  const conn2 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });


  const ConnectCrypto = conn2.model("Coin", connectCryptoSchema);


  module.exports = connectCryptoSchema;



  ConnectCrypto.find({user: currentUser},function(err,foundUsers){
    if (!foundUsers) {


      res.render('cryptopayment',{message: '' , connectedCryptos: [] ,
cryptobalance: 0});


    }else{


    res.render('cryptopayment',{message:'' , connectedCryptos: foundUsers ,
cryptobalance: 0});
    }



  });


});
```

```
app.get("/payment", function(req, res) {

  let balance;

  const conn2 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });

  const Wallet = conn2.model("Wallet", walletSchema);

  module.exports = walletSchema;

  Wallet.findOne({user: currentUser},function(err, foundWallet){

    if (!foundWallet) {


    }else {

      balance = foundWallet.balance;

    }

    const ConnectCrypto = conn2.model("Coin", connectCryptoSchema);

    module.exports = connectCryptoSchema;

    ConnectCrypto.find({user: currentUser},function(err,foundUsers){
      if (!foundUsers) {
        res.render('payment',{message: '', balance: balance , connectedCryptos: [] ,
cryptobalance: 0});


      }else{


        res.render('payment',{message: '', balance: balance , connectedCryptos:
foundUsers , cryptobalance: 0});

      }



    });


  });


});



app.get("/transactions", function(req, res) {

  res.sendFile(__dirname + "/transactions.html");

});


app.get("/feedback", function(req, res) {

  res.sendFile(__dirname + "/feedback.html");

});


app.get("/load", function(req, res) {

  res.sendFile(__dirname + "/load.html");

});


app.get("/pin", function(req, res) {

  res.sendFile(__dirname + "/pin.html");
```

```
});



app.post("/login", function(req, res) {

  const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });
  // .catch(error => handleError(error));;

  const User = conn1.model("User", userSchema);

  module.exports = userSchema;

  currentUser = req.body.userName;

  User.findOne({
    username: req.body.userName,
    password: req.body.password
  }, function(err, foundUser) {
    if (!foundUser) {

      res.render('form',{message: "Username/Password does not match! Try logging in
again."});

      // res.send("Username/Password does not match! Try logging in again.")
    } else {
      //  res.sendFile(__dirname+ "/dashboard.html");

      res.redirect("/dashboard");
    }
  });
});



app.post("/signup", function(req, res) {
  const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });

  const User = conn1.model("User", userSchema);

  module.exports = userSchema;

  const user = new User({
    username: req.body.userName,
    email: req.body.email,
    password: req.body.password
  });

  User.findOne({
    $or: [{
      username: req.body.userName
    }, {
      email: req.body.email
    }]
  }, function(err, foundUser) {
    if (!foundUser) {
      user.save();
      res.render('form',{message: "Successfully signed up. Please login to your
account."});


      // res.redirect("/login");

    } else {


      res.render('form',{message: "Username already taken/ Email is already
registered, please choose a different username/use different email."});


      // res.send("Username already taken/ Email is already registered, please
choose a different username/use different email.");


    }

  });


});


// Dashboard BACKEND


//1.Connect accounts



app.post("/bank", function(req, res) {

  const conn2 = mongoose.createConnection("mongodb://localhost: 27017/bankDB", {

    useNewUrlParser: true,

    poolSize: 10

  });


  const Bank = conn2.model("Bank", bankSchema);


  module.exports = bankSchema;
  // console.log(currentUser);


  let bankName = req.body.bankName;

  let accNum = req.body.accountNumber;

  let pw = req.body.password;


  // console.log(bankName, accNum, pw);


  Bank.findOne({

    bankname: bankName,

    accountnumber: accNum,

    password: pw

  }, function(err, foundUser) {


    // console.log(foundUser);


    if (!foundUser) {

      console.log(foundUser);

      res.send("No user found");

    } else {


      console.log(foundUser);



      const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB",
{

        useNewUrlParser: true,

        poolSize: 10

      });

      // .catch(error => handleError(error));;
```

```
const ConnectBank = conn1.model("Connect", connectBankSchema);

module.exports = connectBankSchema;

// console.log(foundUser);
//
//
//
// mongoose.connection.close();

let nameOfbank = foundUser.bankname;
let acN = foundUser.accountnumber;
let passw = foundUser.password;
let bal = foundUser.balance;

console.log(nameOfbank, acN, passw, bal + 1000);

ConnectBank.find({
  bankname: nameOfbank,
  accountnumber: acN
}, function(err, foundAccount) {

  console.log(foundAccount);
  if (foundAccount.length != 0) {

    res.render('connect',{message: "Account already connected."});

    // res.send("Account already connected.");
  } else {

    const bank = new ConnectBank({
            username: currentUser,

            bankname: nameOfbank,

            accountnumber: acN,

            password: passw,

            balance: bal

          });


          bank.save();


          res.render('connect',{message: "Account successfully connected."});


          // res.send("Account successfully connected.");


        }




      });


      //

      // ConnectBank.findOne({bankname: nameOfbank,accountnumber:
foundUser.accountnumber},function(err,foundAccount){

      //

      //   console.log(foundAccount);

      //   if (!foundAccount) {

      //

      //     console.log("NO ACCOUNTS");

      //     console.log(currentUser);

      //

      //

      //

      //     const bank = new ConnectBank({
```

```
      //       username: currentUser,

      //       bankname: nameOfbank,

      //       accountnumber: acN,

      //       password: passw,

      //       balance: bal

      //     });

      //

      //     // ConnectBank.create({bank},function(err,result){

      //     //   if(err){

      //     //     console.log(err);

      //     //   }else{

      //     //     res.send("Account Connected!");

      //     //   }

      //     // })

      //

      //     ConnectBank.findOne({bankname: nameOfbank, accountnumber:
acN},function(err,foundAccount){

      //

      //       console.log(foundAccount);

      //       if (!foundAccount) {

      //

      //             bank.save();

      //

      //

      //

      //             res.send("Account Connected!");

      //             console.log("HI");

      //

      //       }else{

      //

      //         res.send("Same account cannot be connected twice.")

      //       }
      //     });
      //
      //
      //
      //     // User.create({bank: bank},function(err,result){
      //     //   if (err) {
      //     //     console.log(err);
      //     //   }else {
      //     //     console.log("Added bank");
      //     //   }
      //     // })
      //
      //     // User.updateOne({username: currentUser},{$set:{bank: bank}
},function(err){
      //     //   if (err) {
      //     //       console.log(err);
      //     //   }else {
      //     //     console.log("successfully updated");
      //     //   }
      //     // });
      //
      //
      //
      //
      //
      //
      //
      //   }else{
      //     mongoose.connection.close();
      //     res.send("Account already connected!");
      //   }
      // });
      // res.send("Account connected successfully.");
    }
  });


});





app.post("/crypto", function(req, res) {

  const conn3 = mongoose.createConnection("mongodb://localhost: 27017/cryptoDB", {
    useNewUrlParser: true,
    poolSize: 10
  });
  // .catch(error => handleError(error));;

  const Crypto = conn3.model("Crypto", cryptoSchema);



  module.exports = cryptoSchema;



  let cryptoName = req.body.cryptoName;
  let userName = req.body.userNameCrypto;
  let pw = req.body.cryptoPassword;



  Crypto.findOne({
    $and: [{
      cryptoname: cryptoName
    }, {
      username: userName
    }, {
      password: pw
    }]
  }, function(err, foundUser) {



    if (!foundUser) {


        res.render('connect',{message: "No user found"});


      // res.send("No user found");
    } else {
      console.log(foundUser);


      const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB",
{
        useNewUrlParser: true,
        poolSize: 10
      });
      // .catch(error => handleError(error));;


      const ConnectCrypto = conn1.model("Coin", connectCryptoSchema);
```

```
module.exports = connectCryptoSchema;



// console.log(foundUser);
//
//
//
// mongoose.connection.close();

let nameOfcrypto = foundUser.cryptoname;
let usN = foundUser.username;
let passW = foundUser.password;
let bala = foundUser.balance;
let add = foundUser.address;

// ConnectCrypto.findOne({$and:[{cryptoname: nameOfcrypto},{username:
usN}]},function(err,foundAccount)



ConnectCrypto.count({
  $and: [{
    cryptoname: nameOfcrypto
  }, {
    username: usN
  }]
}, function(err, foundAccount) {

  console.log(foundAccount);
  console.log(ConnectCrypto.db.name, ConnectCrypto.collection.name);
  if (foundAccount > 0) {

      res.render('connect',{message: "Account already connected."});

    // res.send("Account already connected.")
  } else {

    const crypto = new ConnectCrypto({
      user: currentUser,
      cryptoname: nameOfcrypto,
      username: usN,
      password: passW,
      balance: bala,
      address: add
    });

    crypto.save();

    res.render('connect',{message:"Account successfully connected."});

    // res.send("Account successfully connected.");

  }



});



// mongoose.connect("mongodb://localhost: 27017/cryptoDB",{poolSize: 100000,
//   useNewUrlParser: true
```

```
      // }).catch(error => handleError(error));;

      //

      //

      //

      // User.updateOne({username: currentUser},{$set:{crypto: foundUser}
},function(err){

      //   if (err) {

      //       console.log(err);

      //   }else {

      //     console.log("successfully updated");

      //   }

      // });

      //

      // mongoose.connection.close();


    }

  });



});


//2.payment




app.post("/cryptopayment", function(req, res) {


  const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {

    useNewUrlParser: true,

    poolSize: 10

  });

  // .catch(error => handleError(error));;
```

```
const ConnectCrypto = conn1.model("Coin", connectCryptoSchema);

module.exports = connectCryptoSchema;

let cryptoName = req.body.selectedCrypto;
let paymentamt = req.body.paymentAmount;
let recipientAddress = req.body.walletAddress;
let pw = req.body.password;
let username = req.body.username;

ConnectCrypto.findOne({
  user: currentUser,
  cryptoname: cryptoName,
  username: username
}, function(err, foundAccount) {

  console.log(foundAccount + " This is connected crypto's data");


  if (foundAccount) {

    if (foundAccount.balance >= paymentamt) {

      var updatedBalance = foundAccount.balance - parseFloat(paymentamt);

      console.log(updatedBalance);

      ConnectCrypto.updateOne({
        user: currentUser,
        cryptoname: cryptoName,
        username: username
      }, {
        balance: updatedBalance
      }, function(err) {
        if (err) {
          console.log(err);
        } else {
          console.log("successfully updated account's balance");
        }
      });

      ConnectCrypto.findOne({
        cryptoname: cryptoName,
        address: recipientAddress
      }, function(err, found) {
        if (found) {

          var updatedBalance2 = found.balance + parseFloat(paymentamt);

          ConnectCrypto.updateOne({
            cryptoname: cryptoName,
            address: recipientAddress
          }, {
            balance: updatedBalance2
          }, function(err) {
            if (err) {
              console.log(err);
            } else {
              console.log("successfully updated account's balance");
            }
          });
          }
        });



        const conn3 = mongoose.createConnection("mongodb://localhost:
27017/cryptoDB", {
          useNewUrlParser: true,
          poolSize: 10
        });
        // .catch(error => handleError(error));;

        const Crypto = conn3.model("Crypto", cryptoSchema);

        module.exports = cryptoSchema;

        Crypto.findOne({
          cryptoname: cryptoName,
          address: recipientAddress
        }, function(err, foundAccount) {

          console.log(foundAccount);

          if (!foundAccount) {

            ConnectCrypto.find({user: currentUser},function(err,foundUsers){
              if (!foundUsers) {

                res.render('cryptopayment',{message: '' , connectedCryptos: [] });

              }else{
```

```
              res.render('cryptopayment',{message:"Transaction could not be carried out. No account found." , connectedCryptos: foundUsers });

              }

            });

            // res.render('cryptopayment',{message:"Transaction could not be carried out. No account found."});

            // res.send("Transaction could not be carried out. No account found.");

            // <%= hello %> res.render("home", { hello: foundAccount.balance});
          } else {

            var newBal = (foundAccount.balance) + parseFloat(paymentamt);

            console.log(newBal + " This is new balance");

            Crypto.updateOne({
              username: foundAccount.username,
              cryptoname: cryptoName
            }, {
              balance: newBal
            }, function(err) {
              if (err) {
                console.log(err);
              } else {
                console.log(foundAccount);
                console.log("Transaction Completed");
                ConnectCrypto.find({user: currentUser},function(err,foundUsers){
                  if (!foundUsers) {

                    res.render('cryptopayment',{message: '' , connectedCryptos: []
});

                  }else{
                    console.log("Transaction Successful.");

                  // res.render('cryptopayment',{message:"Transaction successful.
Your new balance is "+newBal , connectedCryptos: foundUsers });
                  }


                });
                // res.render('payment',{message:"Transaction successful"});
                // res.send("Transaction successful");
              }
            });

          }
        });

        // ************************************************

        Crypto.findOne({
          cryptoname: cryptoName,
          username: username
        }, function(err, foundAccount) {

          if (!foundAccount) {
            console.log("Balance could not be updated.");

            // res.send("Balance could not be updated.");


            // <%= hello %> res.render("home", { hello: foundAccount.balance});

          } else {


            var newBal2 = (foundAccount.balance) - parseFloat(paymentamt);


            console.log(newBal2 + " This is new balance");


            Crypto.updateOne({

              cryptoname: cryptoName,

              username: username

            }, {

              balance: newBal2

            }, function(err) {

              if (err) {

                console.log(err);

              } else {


                ConnectCrypto.find({user: currentUser},function(err,foundUsers){

                  if (!foundUsers) {


                    res.render('cryptopayment',{message: '' , connectedCryptos: []
});


                  }else{

                    console.log("Transaction Successful.");

                      res.render('cryptopayment',{message:"Transaction successful.
Your new balance is "+newBal2 , connectedCryptos: foundUsers });
```

```
                  // res.render('cryptopayment',{message:"Transaction successful.
Your new balance is "+newBal , connectedCryptos: foundUsers });

                  }



                });


              }

            });

          }

        });



      } else {


        ConnectCrypto.find({user: currentUser},function(err,foundUsers){

          if (!foundUsers) {


            res.render('cryptopayment',{message: '' , connectedCryptos: [] });


          }else{


          res.render('cryptopayment',{message:"Insufficient Balance." ,
connectedCryptos: foundUsers });

          }



        });


        // res.render('payment',{message:"Insufficient Balance."});


        // res.send("Insufficient Balance.")
```

```
   }


  } else {
   console.log(err, "ERROR FOUND");

   ConnectCrypto.find({user: currentUser},function(err,foundUsers){
    if (!foundUsers) {

     res.render('cryptopayment',{message: '' , connectedCryptos: [] });

    }else{

    res.render('cryptopayment',{message:"The account you wanted to pay with is
not linked to your account. " , connectedCryptos: foundUsers });
    }



   });


   // res.render('payment',{message:"The account you wanted to pay with is not
linked to your account. "});
   // res.send("The account you wanted to pay with is not linked to your account.
");
  }
 });

 console.log(cryptoName);

});


app.post("/pin", function(req, res) {
```

```
const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
  useNewUrlParser: true,
  poolSize: 10
});
// .catch(error => handleError(error));;

const Wallet = conn1.model("Wallet", walletSchema);

module.exports = walletSchema;

let user = currentUser;
let balance = 0;
let pin = req.body.transactionPin;

Wallet.find({
  user: user
}, function(err, foundAccount) {

  console.log(foundAccount);
  if (foundAccount.length != 0) {
      res.render('pin',{message:"Wallet already exists."});

    // res.send("Wallet already exists.")
  } else {

    const wallet = new Wallet({
      user: user,
      balance: 0,
      transactionPin: pin
    });
```

```
      wallet.save();


      res.render('pin',{message:"Wallet created successfully."});
      // res.send("Wallet created successfully.");


    }
  });

});




app.post("/load", function(req, res) {

  const conn1 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });
  // .catch(error => handleError(error));;


  const ConnectBank = conn1.model("Connect", connectBankSchema);


  module.exports = connectBankSchema;


  let user = currentUser;
  let bank = req.body.bank;
  let accNumber = req.body.accnum;
  let password = req.body.password;


  let amount = parseFloat(req.body.amount);
  ConnectBank.findOne({
    bankname: bank,
    accountnumber: accNumber,
    username: user,
    password: password
  }, function(err, foundAccount) {

    console.log(foundAccount + "This is from ConnectBank");
    if (!foundAccount) {

        res.render('load',{message:"This bank account is not connected to your
TradeIT account."});

      // res.send("This bank account is not connected to your TradeIT account.")
    } else {

      if (foundAccount.balance < amount) {

          res.render('load',{message:"Not enough balance in your bank account,
cannot load money into Wallet."});
        // res.send("Not enough balance in your bank account, cannot load money into
Wallet.")
      } else {

              console.log("Now we are here!");

              const conn2 = mongoose.createConnection("mongodb://localhost:
27017/usersDB", {
                useNewUrlParser: true,
                poolSize: 10
              });

              const Wallet = conn2.model("Wallet", walletSchema);
```

```
module.exports = walletSchema;


Wallet.findOne({
  user: currentUser
}, function(err, foundWallet) {
  if (!foundWallet) {
    res.render('load',{message:"No wallet found, create wallet
first."});


    // res.send("No wallet found, create wallet first.");
  } else {


    console.log(foundWallet + "This is wallet found");


    var newBalance2 = foundWallet.balance + amount;
    Wallet.updateOne({
      user: currentUser
    }, {
      balance: newBalance2
    }, function(err) {
      if (err) {
        console.log(err);
      } else {
        console.log("Successfully updated wallet's balance");
      }
    });


    var newBalance = foundAccount.balance - amount;


    ConnectBank.updateOne({
      bankname: bank,
      accountnumber: accNumber,
      username: user,
      password: password
    }, {
      balance: newBalance
    }, function(err) {
      if (err) {
        console.log(err);
      } else {

        console.log("Balance updated.");
      }
    });



    const conn3 = mongoose.createConnection("mongodb://localhost:
    27017/bankDB", {
      useNewUrlParser: true,
      poolSize: 10
    });

    const Bank = conn3.model("Bank", bankSchema);

    module.exports = bankSchema;

    console.log(Bank.db.name);

    Bank.findOne({
      bankname: bank,
      accountnumber: accNumber
    }, function(err, foundBankAccount) {
```

```
                    if (!foundBankAccount){
                      console.log("No Account Found ----- Bank");
                    } else {

                      var updatedBalance = parseFloat(foundBankAccount.balance) -
amount;
                      console.log(updatedBalance);

                      Bank.updateOne({
                        bankname: bank,
                        accountnumber: accNumber
                      }, {
                        balance: updatedBalance
                      }, function(err) {

                        if (err) {

                          console.log(err);

                        } else {

                          console.log("Amount updated in Bank's Database too.");
                        }

                      });

                    }

                  });
                  var today = new Date();

                  var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-
'+today.getDate();

                  var time = today.getHours() + ":" + today.getMinutes() + ":" +
today.getSeconds();

                  var dateTime = date+' '+time;

                  console.log(dateTime);

                  const conn4 = mongoose.createConnection("mongodb://localhost:
27017/usersDB", {
                    useNewUrlParser: true,
                    poolSize: 10
                  });

                  const Transaction = conn4.model("Transaction", transactionSchema);

                  module.exports = transactionSchema;

                  const transaction = new Transaction({
                    user: currentUser,
                    transactionamt: amount,
                    balance: newBalance2,
                    transactiontype: "LOAD",
                    date: date,
                    time: time
                  });

                  transaction.save();
                  res.render('load',{message:"Amount added to your wallet."});
                  // res.send("Amount added to your wallet.");

                }
              });

      }

    }

  });

});


app.post("/bankpayment",function(req,res){

  let amount = parseFloat(req.body.paymentAmount);
  let user = req.body.username;
  let transPin = req.body.pin;

  const conn2 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });

  const Wallet = conn2.model("Wallet", walletSchema);

  module.exports = walletSchema;
  const ConnectCrypto = conn2.model("Coin", connectCryptoSchema);


  module.exports = connectCryptoSchema;


  Wallet.findOne({user: currentUser, transactionPin: transPin}, function(err,
foundUser){


    if (!foundUser) {


      ConnectCrypto.find({user: currentUser},function(err,foundUsers){

        if (!foundUsers) {


          res.render('payment',{message: '', balance: balance });


        }else{


        res.render('payment',{message:"Incorrect transaction Pin / No Wallet
connected to your account. Please check!",balance: foundUser.balance });

        }




      });


      // res.render('payment',{message:"Incorrect transaction Pin / No Wallet
connected to your account. Please check!",balance: 0 , connectedCryptos: [] ,
cryptobalance: 0});

      console.log("Incorrect transaction Pin / No Wallet connected to your account.
Please check!");

    } else {

      if (foundUser.balance < amount) {


        ConnectCrypto.find({user: currentUser},function(err,foundUsers){

          if (!foundUsers) {
```

```
      res.render('payment',{message: '', balance: balance });

     }else{

     res.render('payment',{message:"NOT ENOUGH BALANCE IN YOUR WALLET. PLEASE
LOAD MONEY!",balance: foundUser.balance });
     }



    });

    // res.render('payment',{message:"NOT ENOUGH BALANCE IN YOUR WALLET. PLEASE
LOAD MONEY!",balance: foundUser.balance , connectedCryptos: [] , cryptobalance: 0});

    // res.send("NOT ENOUGH BALANCE IN YOUR WALLET. PLEASE LOAD MONEY!");
   }else {
    Wallet.findOne({user: user},function(err,foundReceiver){
     if (!foundReceiver) {

      ConnectCrypto.find({user: currentUser},function(err,foundUsers){
       if (!foundUsers) {

        res.render('payment',{message: '', balance: balance });

       }else{

       res.render('payment',{message:"The receiver with the username "+user+"
does not exist.",balance: foundUser.balance });
       }



      });
```

```
          // res.render('payment',{message:"The receiver with the username
"+user+" does not exist.",balance: foundUser.balance , connectedCryptos: [] ,
cryptobalance: 0});



          // res.send("The receiver with the username "+user+" does not exist.");
        }else {

          let newBalance = foundReceiver.balance + amount;
          let newBalance2 = foundUser.balance - amount
          Wallet.updateOne({user: user},{balance: newBalance},function(err){
            if (err) {
              console.log(err);
            }else {
              console.log("Transaction successfull.");


              ConnectCrypto.find({user: currentUser},function(err,foundUsers){
                if (!foundUsers) {

                  res.render('payment',{message: '', balance: balance });

                }else{

                res.render('payment',{message:"Transaction successfull.",balance:
newBalance2 });
                }


              });
```

```
                // res.render('payment',{message:"Transaction successfull.",balance:
newBalance2 , connectedCryptos: [] , cryptobalance: 0});

              }
            });

            Wallet.updateOne({user: currentUser, transactionPin: transPin},
{balance: newBalance2 }, function(err){
              if (err) {
                console.log(err);
              }else {
                console.log("Successfully updated the sender's wallet's balance");
              }
            });

            var today = new Date();

            var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-
'+today.getDate();

            var time = today.getHours() + ":" + today.getMinutes() + ":" +
today.getSeconds();

            var dateTime = date+' '+time;

            console.log(dateTime);

            const conn3 = mongoose.createConnection("mongodb://localhost:
27017/usersDB", {
```

```
          useNewUrlParser: true,
          poolSize: 10
        });

        const Transaction = conn3.model("Transaction", transactionSchema);

        module.exports = transactionSchema;

        const transaction = new Transaction({
          user: currentUser,
          transactionamt: amount,
          balance: newBalance2,
          transactiontype: "DEBIT",
          date: date,
          time: time
        });

        transaction.save();

        const transaction2 = new Transaction({
          user: user,
          transactionamt: amount,
          balance: newBalance,
          transactiontype: "CREDIT",
          date: date,
          time: time
        });

        transaction2.save();

      }
```

```
        });
      }
    }
  });

});


app.post("/transactionlog", function(req,res){

  const conn3 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });

  const Transaction = conn3.model("Transaction", transactionSchema);

  module.exports = transactionSchema;

Transaction.find({user: currentUser},function(err,foundTransactions){

      if(foundTransactions.length != 0 ){

        res.render("transactions",{message: "You can view the transaction log
below.", transactions: foundTransactions,transactionGraph: []});

      }else {

        res.render('transactions',{message: "No transactions found!", transactions:
[],transactionGraph: []});

      }
```

```
});

});

app.post("/transactiongraph",function(req,res){

  const conn3 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {
    useNewUrlParser: true,
    poolSize: 10
  });

  const Transaction = conn3.model("Transaction", transactionSchema);

  module.exports = transactionSchema;

Transaction.find({user: currentUser},function(err,foundTransactions){

  var dataArray = [];

  foundTransactions.forEach(function(transaction){
   var array = [];
   array.push(transaction.date);
   array.push(transaction.balance);

   console.log(array);

  dataArray.push(array);

 });
```

```
console.log(dataArray);


      if(foundTransactions.length != 0 ){


        res.render("transactions",{message: "You can view the graph below.",
transactions: [],transactionGraph: dataArray});


      }else {


        res.render('transactions',{message: "No transactions found!", transactions:
[],transactionGraph: []});


      }


});


// res.render('transactions',{message:"You can see the graph",transactions:[],graph:
"image", transactionGraph: ["hi",123]});


});



app.post("/feedback", function(req,res){


    let rating = req.body.stars;

    let feedback1 = req.body.feedback;


    const conn3 = mongoose.createConnection("mongodb://localhost: 27017/usersDB", {

      useNewUrlParser: true,

      poolSize: 10

    });
    const Feedback = conn3.model("Feedback", feedbackSchema);


    module.exports = feedbackSchema;


    const feedback = new Feedback({
      user: currentUser,
      rating: rating,
      feedback: feedback1
    });


    feedback.save();


    res.render('feedback',{message: "Thank you for your precious feedback!"});



});



function handleError(error) {
  console.log(error);
}


app.listen(3000, function() {
  console.log("Server started on port 3000");
});
```

RESULTS:

The screenshot of the results has already been attached above.

The screenshot of the results includes the successful linking of the bank account, loading the wallet, connecting crypto account, making payments through the crypto and the digital wallet and the transaction history was shown in a table.

TEST REPORT:

We were successfully able to implement and use all of the features that we had included on the each of the pages, basically website. We were encountering some issues while connecting and making payment via the crypto account and wallet. But then we fixed that error and then everything is working fine now. But still, improvements can be done on this project on which we will work in the future.

ADVANTAGES:

- It consists of making digital transaction using both the digital and the crypto wallet.
- Using our website will allow people to make transactions by staying wherever they are and people need not to travel to bank or anywhere for the money trading. Eventually leading to the minimization of cost of time and money.
- People can make payment securely.

CONCLUSION:

We would like to conclude this project by saying that we have successfully implemented it and achieved our objective. The online currency trading using digital wallet was the title of the project and accordingly we tried to include all the features required for a completeness of the website. This system can now be implemented in any system throughout the world. Throughout this process we also learnt the usage of various technologies like JS and mongoDB to work on.

We would like to thank **Prof.Dr.Mythili T** Ma’am for providing us this esteemed opportunity to showcase our skills. The future scope of this project includes the further enhancements we can make in order to make it more feasible. Firstly, we would like to ensure further projection maybe hardware driven devices like an iris scanner or fingerprint system. Secondly, we would also probably like to integrate dbms with secure cloud services to ensure the proper backup of information. Lastly, we will aim for multilingual support in order to sever a larger audience globally.

REFERENCES

Class notes and referential materials provided in course CSE3002 by Prof.Dr.Mythili T.

[1] Steven Holzner, “HTML Black Book”, Jon Skeet,”C# in depth

[2] https://www.tutorialspoint.com/mongodb/index.htm
[3] https://www.w3schools.com/js/