PES UNIVERSITY
Software Engineering
Case Study – 04
Lab Tasks

ATM Simulator

Contributed by:-

1) Tushar J (PES1UG20CS472)

2)Prem Sagar J S (PES1UG20CS825)

3)Vaibhav Vijay (PES1UG20CS479)

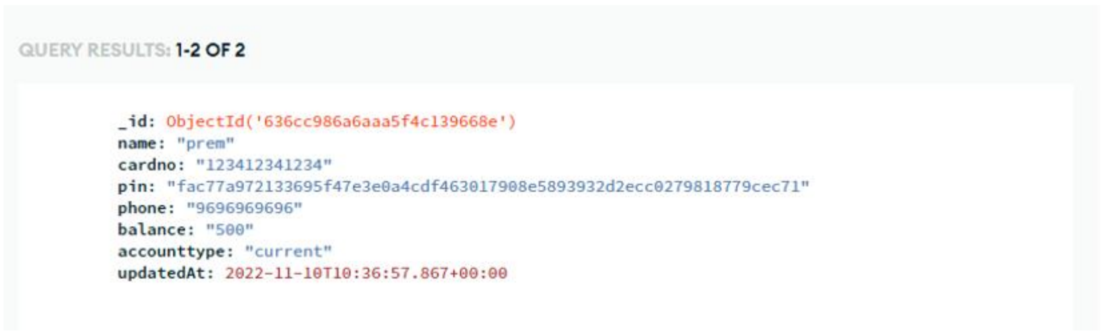
4)Thushar M (PES1UG20CS471)

Problem statement-1 (Unit Testing):-

Unit description : when user gives card number and pin , this API will fetches/queries the DB for the user info which is available on the DB and fetches and displays in the resposnse page.

```
exports.AccountInfo = async (req, res) => {
    const {cardno , pin} = req.body;
    // fetching in DB
    const user = await User.findOne({
        cardno,
        pin: sha256(pin+process.env.SALT),
    });
    if(!user) throw "Invalid Pin !";
    res.json({
        message: "Account Info has fetched",
        account: user, // this user is JSON Object  which contains user name, pin, balance, card no, account type, phone number can be extracted in front end

    });

};
```

Available data in the DB:-

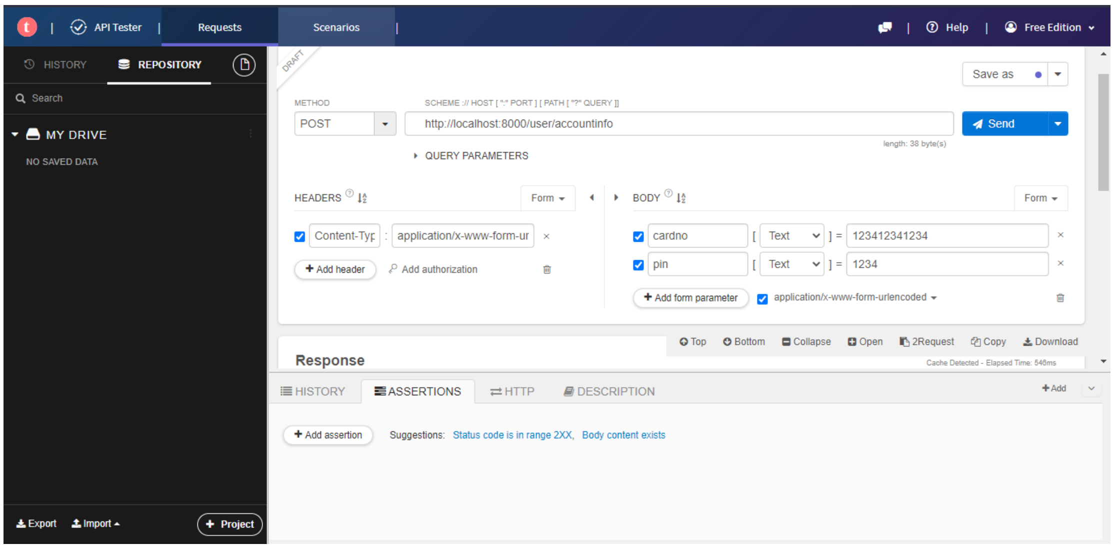


QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('636cc986a6aaa5f4c139668e')
name: "prem"
cardno: "123412341234"
pin: "fac77a972133695f47e3e0a4cdf463017908e5893932d2ecc0279818779cec71"
phone: "9696969696"
balance: "500"
accounttype: "current"
updatedAt: 2022-11-10T10:36:57.867+00:00
```

Above is expected data to be fetched by the API

We are using TELEND API TESTER Tool for testing our API's



METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8000/user/accountinfo

length: 38 byte(s)

HEADERS: Content-Typ : application/x-www-form-ur

BODY:

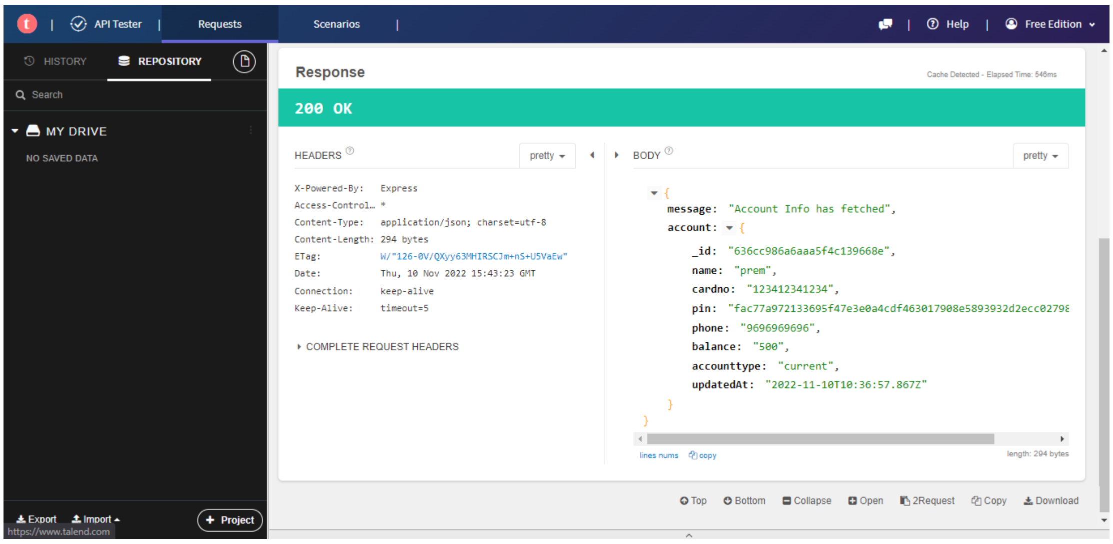
cardno	Text	123412341234
pin	Text	1234

application/x-www-form-urlencoded

Response

Cache Detected - Elapsed Time: 546ms

Suggestions: Status code is in range 2XX, Body content exists



Response

Cache Detected - Elapsed Time: 546ms

200 OK

HEADERS

```
X-Powered-By:    Express
Access-Control…  *
Content-Type:    application/json; charset=utf-8
Content-Length:  294 bytes
ETag:            W/"126-0V/QXyy63MHIRSCJm+nS+U5VaEw"
Date:            Thu, 10 Nov 2022 15:43:23 GMT
Connection:      keep-alive
Keep-Alive:      timeout=5
```

COMPLETE REQUEST HEADERS

BODY

```
{
  message:  "Account Info has fetched",
  account:  {
    _id:  "636cc986a6aaa5f4c139668e",
    name:  "prem",
    cardno:  "123412341234",
    pin:  "fac77a972133695f47e3e0a4cdf463017908e5893932d2ecc0279
    phone:  "9696969696",
    balance:  "500",
    accounttype:  "current",
    updatedAt:  "2022-11-10T10:36:57.867Z"
  }
}
```

length: 294 bytes

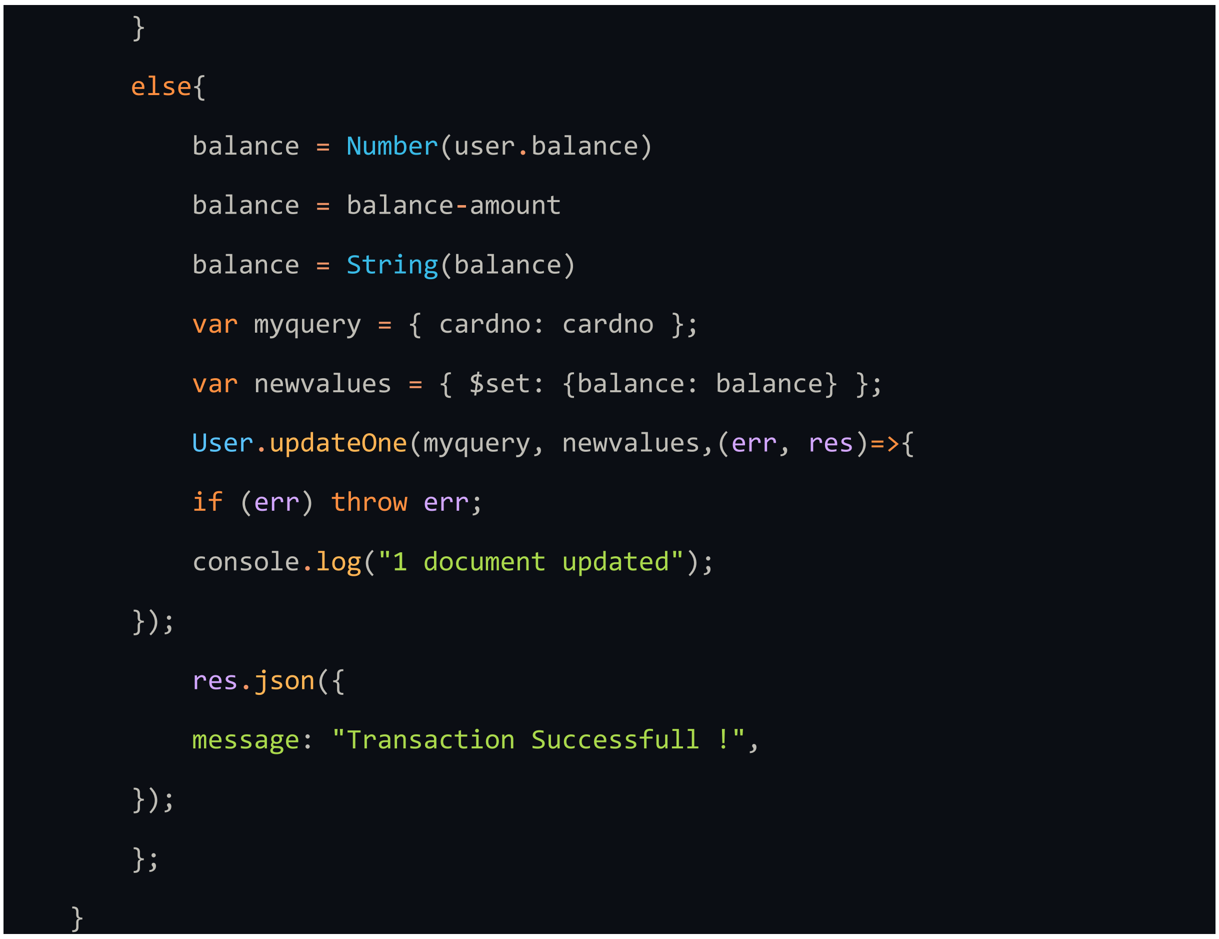
Problem Statement-2 (Dynamic Testing):-

2.a) Boundary Value Analysis:-

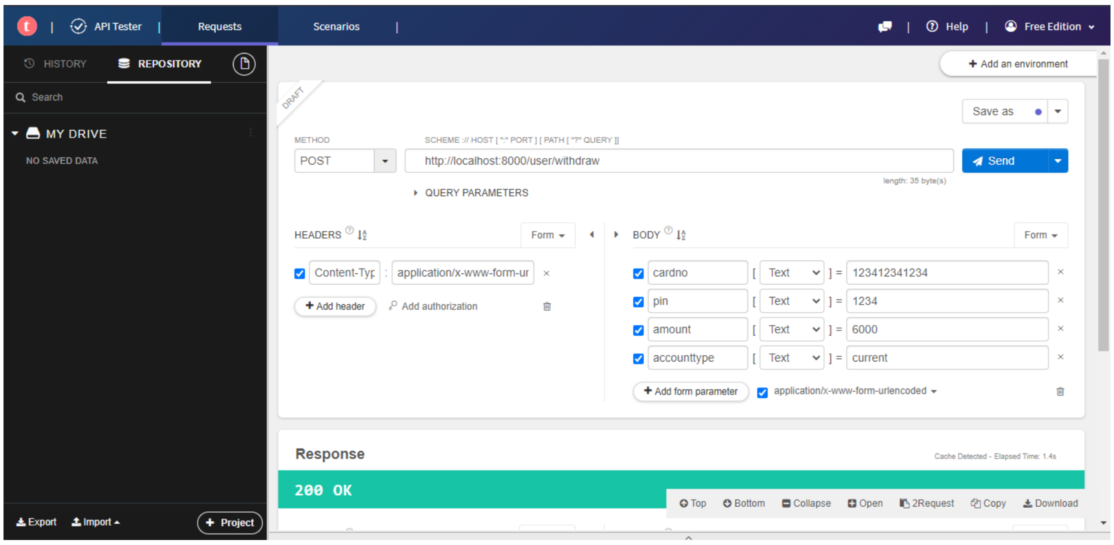
This is a technique where we analyze the behavior of the application with test data residing at the boundary values of the equivalence classes.

By using the test data residing at the boundaries, there is a higher chance of finding errors in the software application.

```
exports.withdrawMoney = async (req, res) => {

    const {cardno , pin , amount, accounttype} = req.body;

    const user = await User.findOne({

        cardno,

        pin: sha256(pin+process.env.SALT),

    });

    if(!user) throw "Invalid Pin";

    if(accounttype==user.accounttype){

        if(amount>Number(user.balance)){

            res.json({

                message: "Insufficient Funds !"

            });
```



```
        }
        else{
            balance = Number(user.balance)
            balance = balance-amount
            balance = String(balance)
            var myquery = { cardno: cardno };
            var newvalues = { $set: {balance: balance} };
            User.updateOne(myquery, newvalues,(err, res)=>{
            if (err) throw err;
            console.log("1 document updated");
        });
            res.json({
            message: "Transaction Successfull !",
        });
        };
    }
```



METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8000/user/withdraw
length: 35 byte(s)

HEADERS

✓	Content-Typ	application/x-www-form-ur

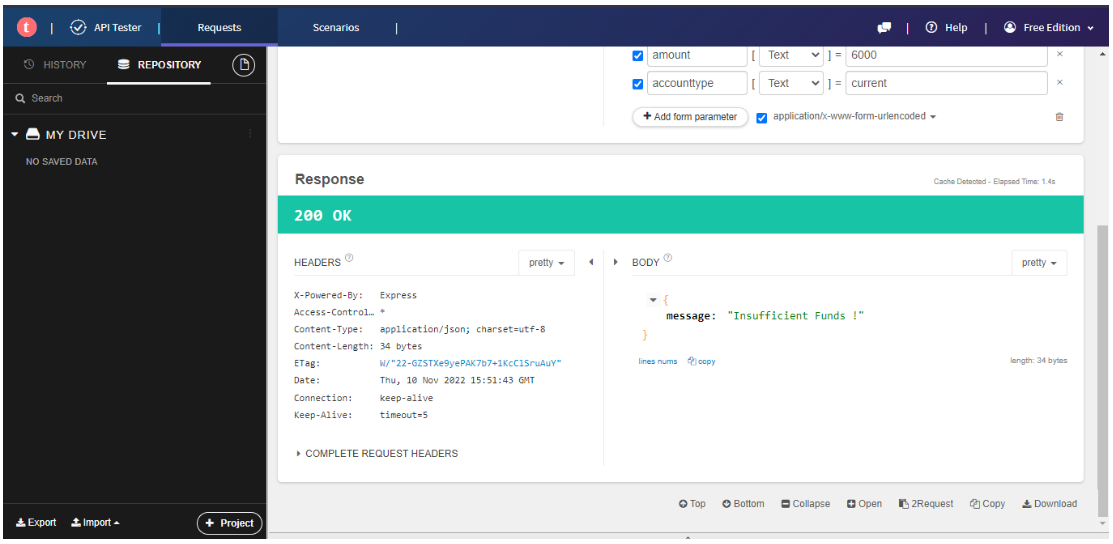
BODY

✓	cardno	Text	123412341234
✓	pin	Text	1234
✓	amount	Text	6000
✓	accounttype	Text	current

✓ application/x-www-form-urlencoded

Response
Cache Detected - Elapsed Time: 1.4s

200 OK


amount [Text] = 6000

accounttype [Text] = current

application/x-www-form-urlencoded

Response

Cache Detected - Elapsed Time: 1.4s

200 OK

HEADERS

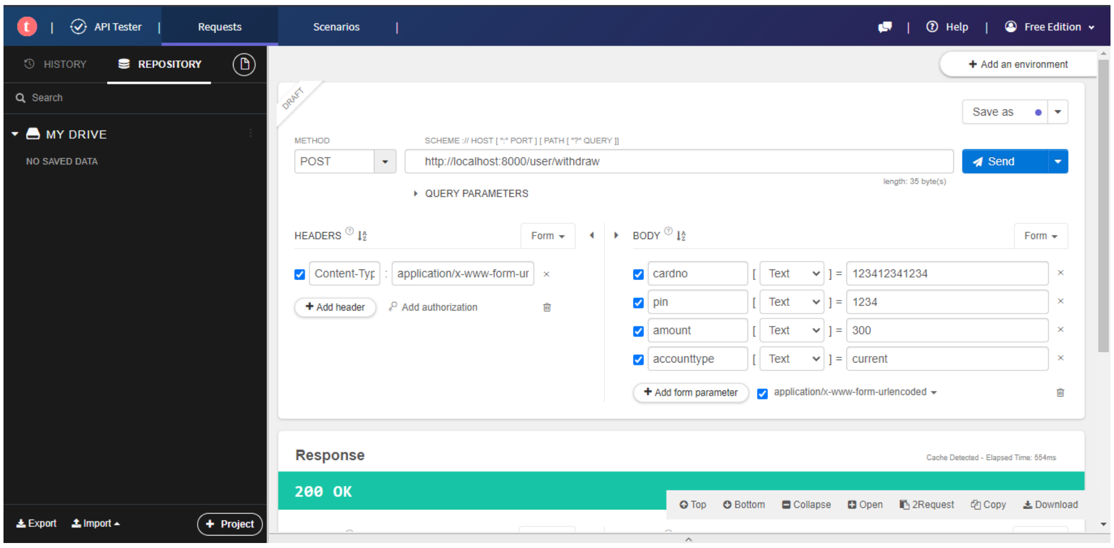
```
X-Powered-By:   Express
Access-Control… *
Content-Type:   application/json; charset=utf-8
Content-Length: 34 bytes
ETag:           W/"22-GZSTXe9yePAK7b7+1KcClSruAuY"
Date:           Thu, 10 Nov 2022 15:51:43 GMT
Connection:     keep-alive
Keep-Alive:     timeout=5
```

BODY

```
{
  message: "Insufficient Funds !"
}
```

length: 34 bytes

Valid :


METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8000/user/withdraw

length: 35 byte(s)

HEADERS

Content-Typ : application/x-www-form-ur

BODY

cardno [Text] = 123412341234

pin [Text] = 1234

amount [Text] = 300

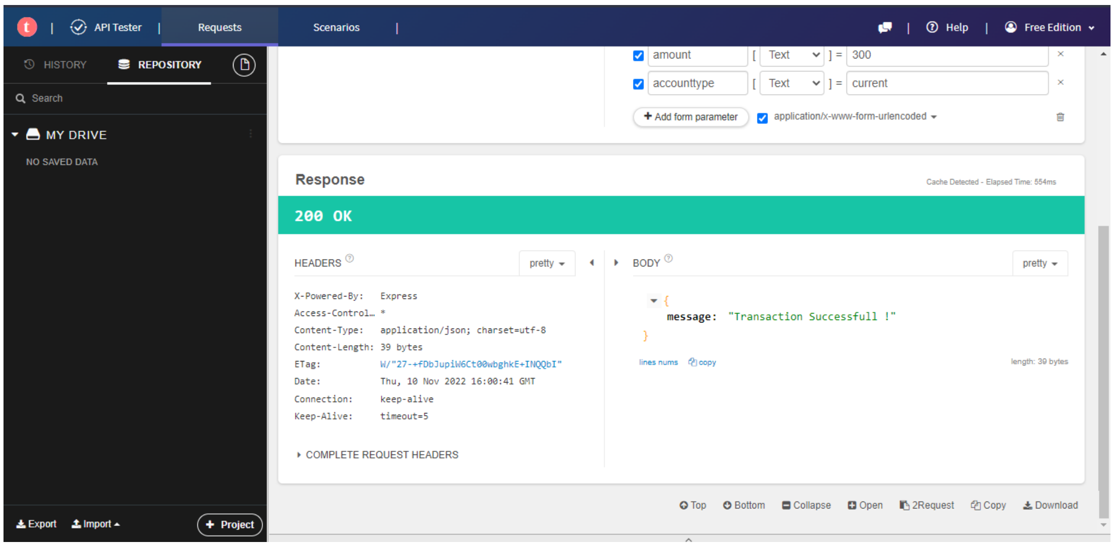
accounttype [Text] = current

application/x-www-form-urlencoded

Response

Cache Detected - Elapsed Time: 554ms

200 OK


amount [Text] = 300

accounttype [Text] = current

application/x-www-form-urlencoded

Response

Cache Detected - Elapsed Time: 554ms

200 OK

HEADERS

```
X-Powered-By:   Express
Access-Control… *
Content-Type:   application/json; charset=utf-8
Content-Length: 39 bytes
ETag:           W/"27-+fDbJupiW6Ct00wbghkE+INQQbI"
Date:           Thu, 10 Nov 2022 16:00:41 GMT
Connection:     keep-alive
Keep-Alive:     timeout=5
```

BODY

```
{
  message: "Transaction Successfull !"
}
```

length: 39 bytes

2.b) Mutation Testing:-

Here we have taken three mutation of withdrawMoney function.

Mutantion-1:-

In this mutant, user is able to proceed with transaction and also get access to thing inside even after pin he entered is wrong because here we are not checking for pin at all. So lets us not to remove it from original copy of code.// thus making it one boundary condition.

```
exports.withdrawMoney = async (req, res) => {
    const {cardno , pin , amount, accounttype} = req.body;
    const user = await User.findOne({
        cardno,
        pin: sha256(pin+process.env.SALT),
    });

    if(accounttype==user.accounttype){
        if(amount>Number(user.balance)){
            res.json({
                message: "Insufficient Funds !"
            });
        }
        else{
            balance = Number(user.balance)
            balance = balance-amount
            balance = String(balance)
            var myquery = { cardno: cardno };
            var newvalues = { $set: {balance: balance} };
            User.updateOne(myquery, newvalues,(err, res)=>{
            if (err) throw err;
            console.log("1 document updated");
        });
            res.json({
            message: "Transaction Successfull !",
        });
        };
    }
```

Mutation-2:-

Here in this mutation we have taken down one of our if statement which used to check for users account type, which ensures user account type match for further transaction to happen. For example: a user having savings bank account can't withdraw from business bank account.

Which make it to be there as a one of the condition.

```
exports.withdrawMoney = async (req, res) => {
    const {cardno , pin , amount, accounttype} = req.body;
    const user = await User.findOne({
        cardno,
        pin: sha256(pin+process.env.SALT),
    });
    if(!user) throw "Invalid Pin";

        if(amount>Number(user.balance)){
            res.json({
                message: "Insufficient Funds !"
            });
        }
        else{
            balance = Number(user.balance)
            balance = balance-amount
            balance = String(balance)
            var myquery = { cardno: cardno };
            var newvalues = { $set: {balance: balance} };
            User.updateOne(myquery, newvalues,(err, res)=>{
            if (err) throw err;
            console.log("1 document updated");
        });
```

```
            res.json({
            message: "Transaction Successfull !",
        });
    }
```

Mutation-3:-

Now think if some user trying to withdraw money more then his savings , at this scenario he must get message stating insufficient balance instead of providing money without boundary value checking or analysis on his account savings which leads us to put another if statement to check for that.

```
exports.withdrawMoney = async (req, res) => {
    const {cardno , pin , amount, accounttype} = req.body;
    const user = await User.findOne({
        cardno,
        pin: sha256(pin+process.env.SALT),
    });
    if(!user) throw "Invalid Pin";
    if(accounttype==user.accounttype){
        if(amount>Number(user.balance)){
            res.json({
                message: "Insufficient Funds !"
            });
        }
        else{
            balance = Number(user.balance)
            balance = balance-amount
            balance = String(balance)
            var myquery = { cardno: cardno };
            var newvalues = { $set: {balance: balance} };
            User.updateOne(myquery, newvalues,(err, res)=>{
            if (err) throw err;
            console.log("1 document updated");

        });

            res.json({

            message: "Transaction Successfull !",

        });

        };
    }
```

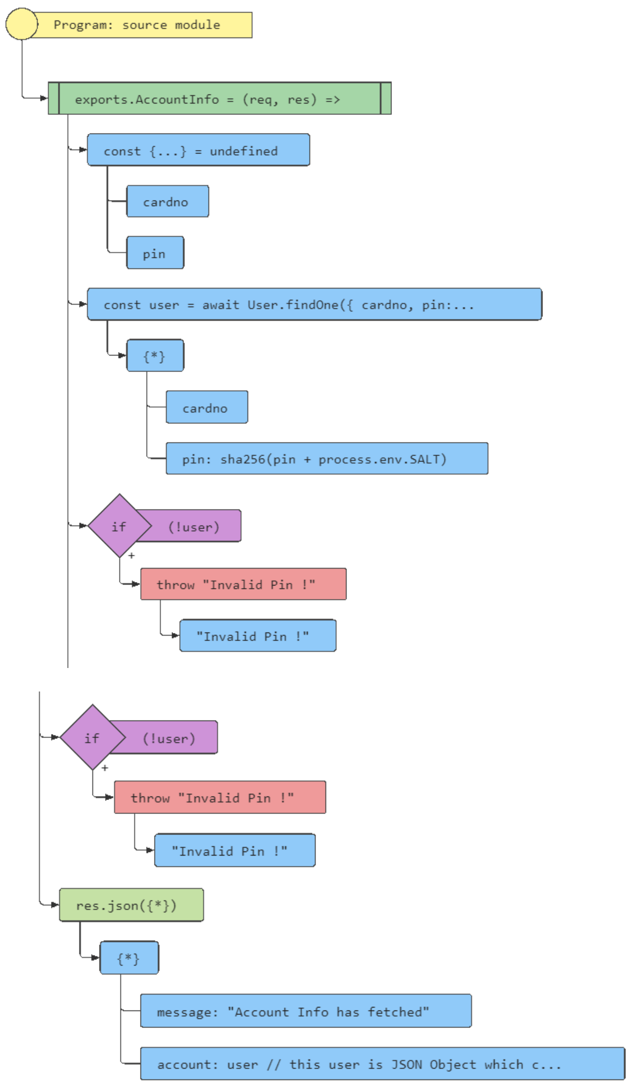
Problem Statement-3 (Static Testing):-

Unit description : when user gives card number and pin , this API will fetches/queries the DB for the user info which is available on the DB and fetches and displays in the resposnse page

Code :

```
exports.AccountInfo = async (req, res) => {

    const {cardno , pin} = req.body;

    // fetching in DB

    const user = await User.findOne({

        cardno,

        pin: sha256(pin+process.env.SALT),

    });


    if(!user) throw "Invalid Pin !";


    res.json({

        message: "Account Info has fetched",

        account: user, // this user is JSON Object  which contains user name,
pin, balance, card no, account type, phone number can be extracted in front
end



    });
};
```

Control Flow Graph:-



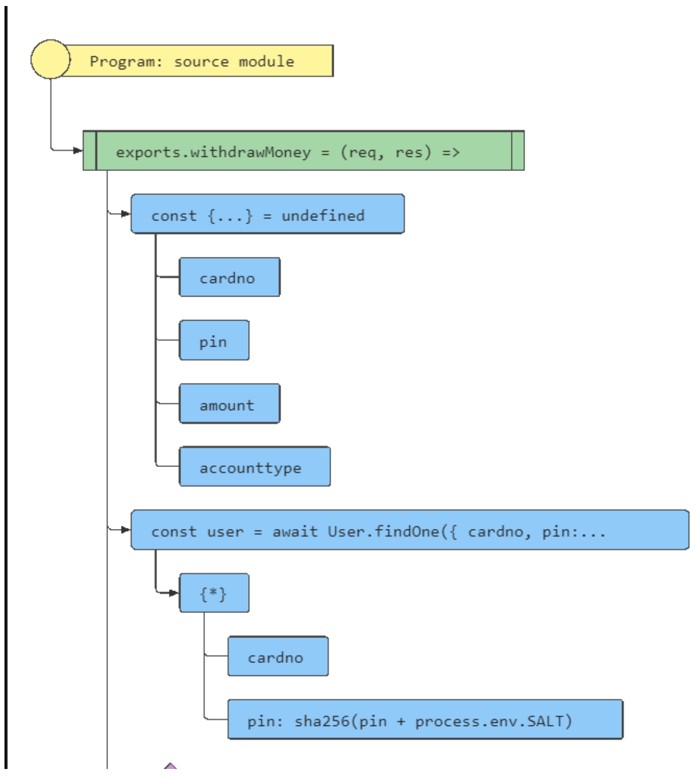
Program: source module
exports.AccountInfo = (req, res) =>
const {...} = undefined
cardno
pin
const user = await User.findOne({ cardno, pin:...
{*}
cardno
pin: sha256(pin + process.env.SALT)
if
(!user)
+
throw "Invalid Pin !"
"Invalid Pin !"
if
(!user)
+
throw "Invalid Pin !"
"Invalid Pin !"
res.json({*})
{*}
message: "Account Info has fetched"
account: user // this user is JSON Object which c...

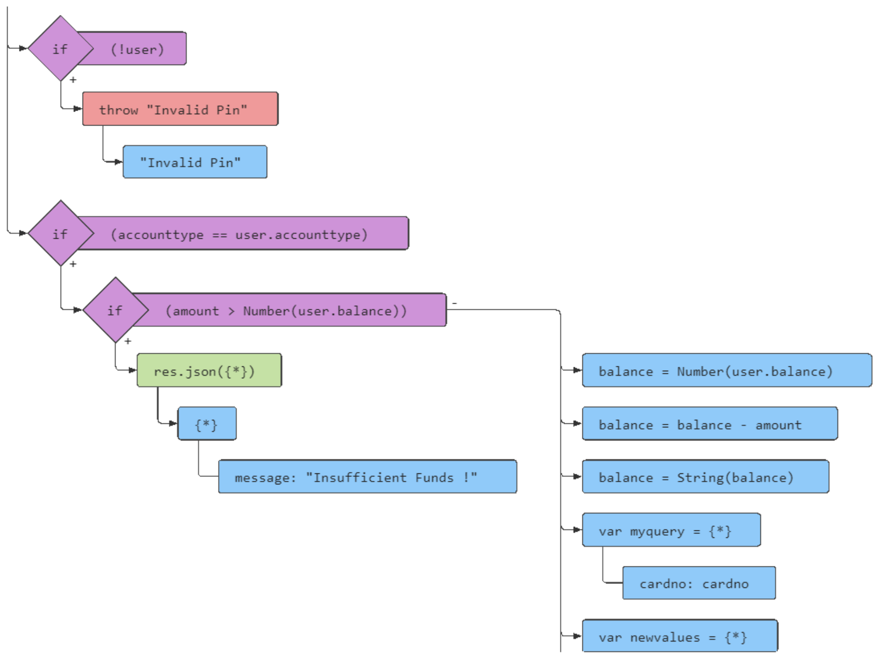
Cyclomatic Complexity=2

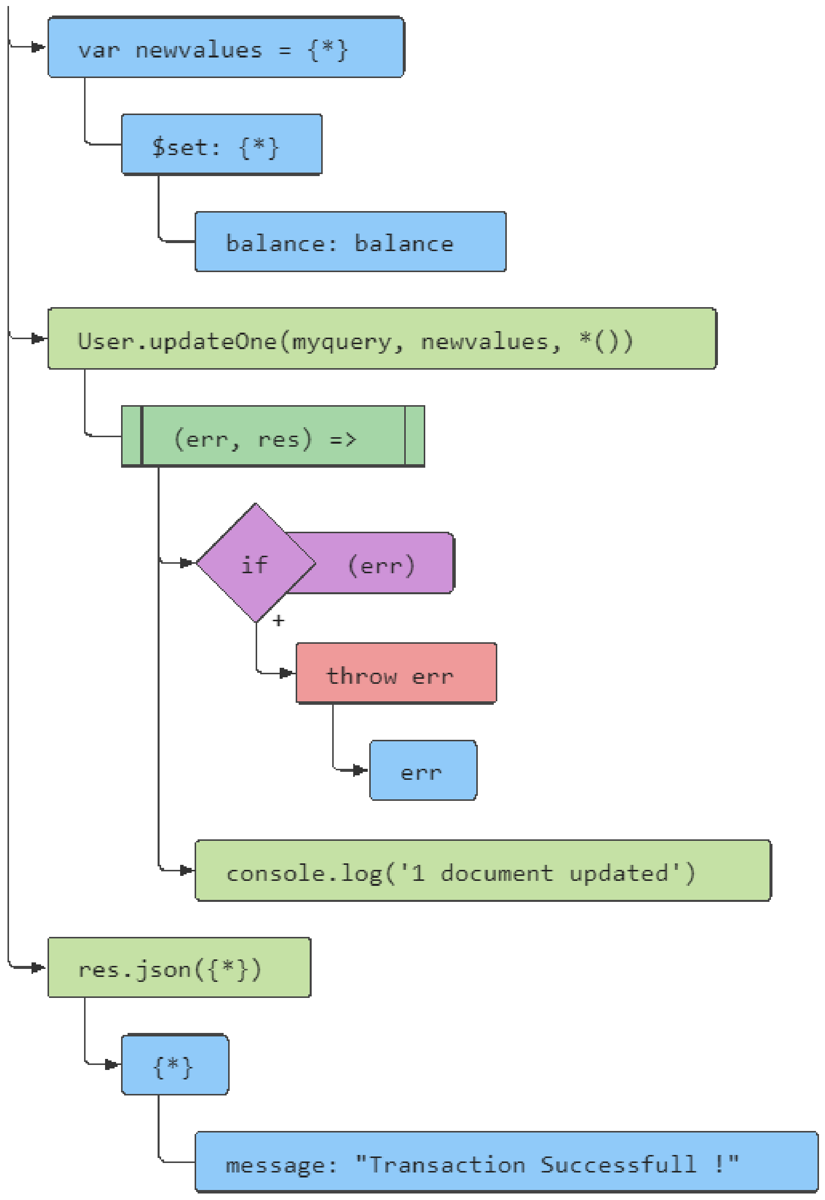
2.

```
exports.withdrawMoney = async (req, res) => {
```

```
    const {cardno , pin , amount, accounttype} = req.body;
    const user = await User.findOne({
        cardno,
        pin: sha256(pin+process.env.SALT),
    });
    if(!user) throw "Invalid Pin";

    if(accounttype==user.accounttype){

        if(amount>Number(user.balance)){
            res.json({
                message: "Insufficient Funds !"
            });
        }
        else{

            balance = Number(user.balance)
            balance = balance-amount
            balance = String(balance)
            var myquery = { cardno: cardno };
            var newvalues = { $set: {balance: balance} };

            User.updateOne(myquery, newvalues,(err, res)=>{
            if (err) throw err;
            console.log("1 document updated");

        });

            res.json({
            message: "Transaction Successfull !",
        });

        };

    }
```


Program: source module
exports.withdrawMoney = (req, res) =>
const {...} = undefined
cardno
pin
amount
accounttype
const user = await User.findOne({ cardno, pin:...
{*}
cardno
pin: sha256(pin + process.env.SALT)


if
(!user)
+
throw "Invalid Pin"
"Invalid Pin"
if
(accounttype == user.accounttype)
+
if
(amount > Number(user.balance))
+
-
res.json({*})
{*}
message: "Insufficient Funds !"
balance = Number(user.balance)
balance = balance - amount
balance = String(balance)
var myquery = {*}
cardno: cardno
var newvalues = {*}


var newvalues = {*}
$set: {*}
balance: balance
User.updateOne(myquery, newvalues, *())
(err, res) =>
if
(err)
+
throw err
err
console.log('1 document updated')
res.json({*})
{*}
message: "Transaction Successfull !"

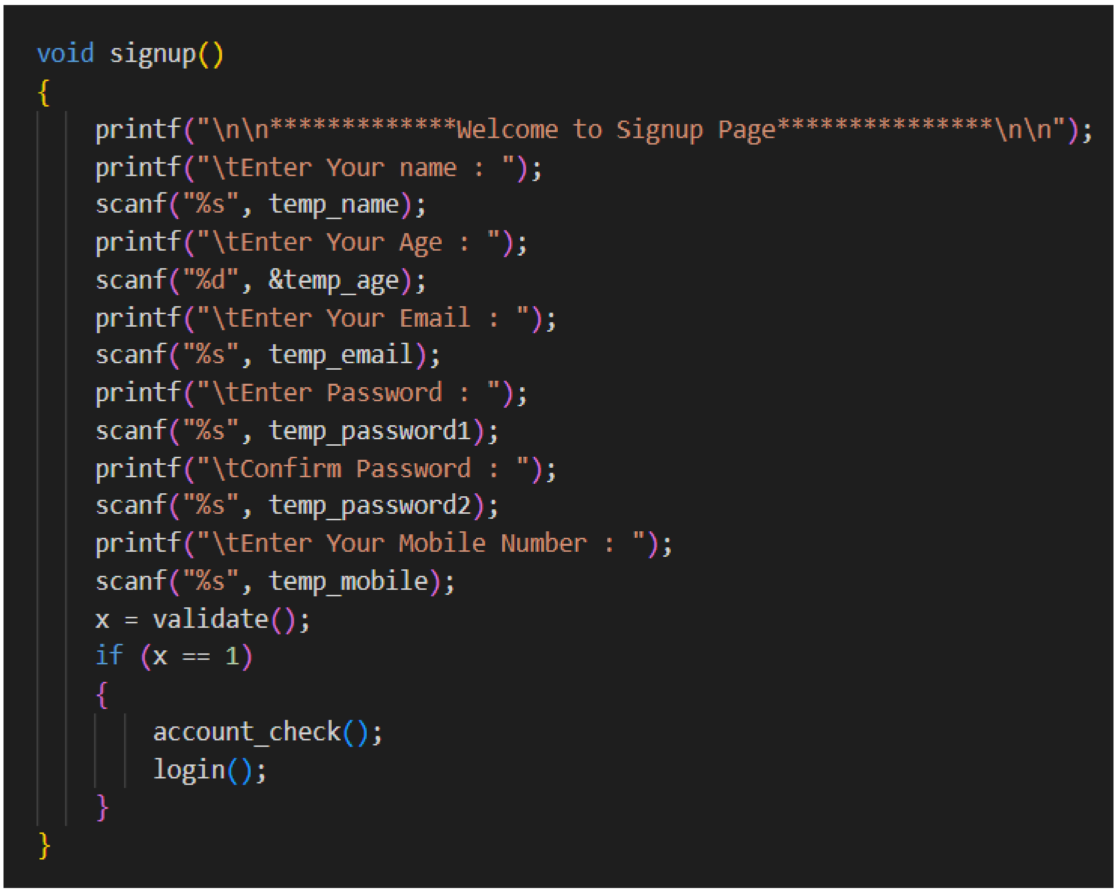
Problem Statement-4 (Acceptance Testing):-

Assume your neighboring team is the client for your code. Give them an idea of what your product is and the software requirements for the product.

• Exchange your code base and test each others projects to see if it meets user requirements

• If you identify a bug in the project you are testing, inform the opposing team of the bug

• As a team, based in clients experience, ideate modifications to the existing project that could improve client experience



```c
void signup()
{
    printf("\n\n*************Welcome to Signup Page***************\n\n");
    printf("\tEnter Your name : ");
    scanf("%s", temp_name);
    printf("\tEnter Your Age : ");
    scanf("%d", &temp_age);
    printf("\tEnter Your Email : ");
    scanf("%s", temp_email);
    printf("\tEnter Password : ");
    scanf("%s", temp_password1);
    printf("\tConfirm Password : ");
    scanf("%s", temp_password2);
    printf("\tEnter Your Mobile Number : ");
    scanf("%s", temp_mobile);
    x = validate();
    if (x == 1)
    {
        account_check();
        login();
    }
}
```

Checking if Name entered is valid or invalid

Test Case passes if Name has only letters and fails if Name has digits or special characters

No	Input	Output	Test Case
1	Tushar123	Invalid Name	Fail
2	Vaibhav Vijay	Valid Name	Pass

Checking if Age entered is valid or invalid

Test Case passes if Age > 0 and fails if age <= 0

No	Input	Output	Test Case
1	-5	Invalid Age	Fail
2	18	Valid Age	Pass

Checking if Email entered is valid or invalid

Test Case passes if Email has domain name, @ and dot. Email can have 2 dots if it has sub domain. Email can have special characters. Email can have square bracket around IP address. Email can have digits. Test Case fails if Email does not have above mentioned features

No	Input	Output	Test Case
1	tushar.jumlagmailco	Invalid Email	Fail

2	tushar.jumla@gmail.com	Valid Email	Pass

Checking if Password entered is valid or invalid

Test Case passes if Password is strong and fails if Password is weak

No	Input	Output	Test Case
1	vv22	Invalid Password	Fail
2	vaiBhaV@25	Valid Password	Pass

Checking if Password entered is correct or wrong

Test Case passes if Password matches with initial Password and fails if Password does not matches with initial Password

No	Input	Output	Test Case
1	VaibhaV@25	Wrong Password	Fail
2	vaibhaV@25	Correct Password	Pass

Checking if entered Phone Number entered is correct or wrong

Test Case passes if Phone Number has only digits and length is 10. Test Case fails if Phone Number has letters or special characters and length is not equal to 10

No	Input	Output	Test Case
1	983902031	Invalid Phone Number	Fail
2	9929898939	Valid Phone Number	Pass

Problem Statement-5 (Maintenance Activities):-

Once a product is completed, it is handed off to a service based company to ensure all maintenance activities are performed without the added expenditure of skilled developers. However, a few tasks are performed by the maintenance team to gauge the product better. In this problem statement, you will be asked to experiment with your code.

• Exchange code bases with your neighboring teams and reverse engineer a block of code in order to understand it's functionality

• After understanding the code block, Re-Engineer the code

o Ideate how to refactor the code and the portion of the code base you would have to change

o Discuss how the new changes would impact the time and space complexity of the project during execution

• After Reverse Engineering and Re-Engineering the code, perform acceptance testing between the teams

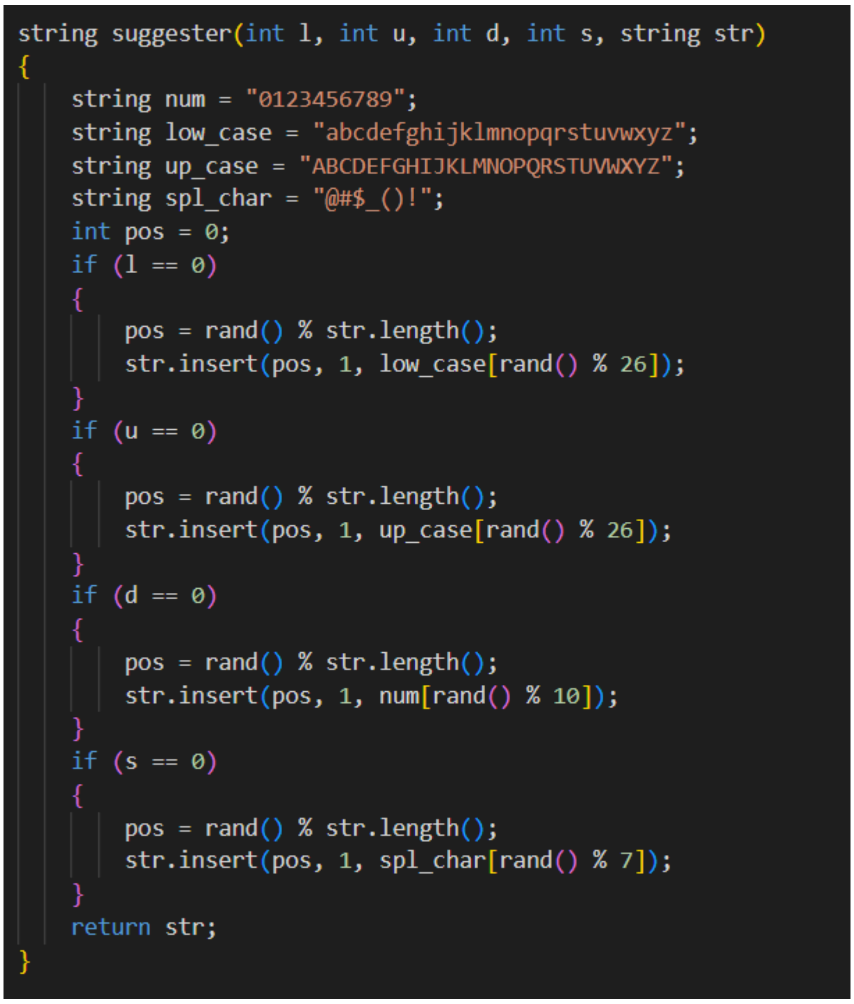
Following code checks whether password entered by user is strong or not.

Strong password has atleast 8 characters, mixture of both uppercase and lowercase letters, mixture of letters and numbers and atleast 1 special character.

If strong password is entered by user then password is validated.

If weak password is entered by user then password is not validated.

When user enters weak password, strong password could be suggested.



```
string suggester(int l, int u, int d, int s, string str)
{
    string num = "0123456789";
    string low_case = "abcdefghijklmnopqrstuvwxyz";
    string up_case = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string spl_char = "@#$_()!";
    int pos = 0;
    if (l == 0)
    {
        pos = rand() % str.length();
        str.insert(pos, 1, low_case[rand() % 26]);
    }
    if (u == 0)
    {
        pos = rand() % str.length();
        str.insert(pos, 1, up_case[rand() % 26]);
    }
    if (d == 0)
    {
        pos = rand() % str.length();
        str.insert(pos, 1, num[rand() % 10]);
    }
    if (s == 0)
    {
        pos = rand() % str.length();
        str.insert(pos, 1, spl_char[rand() % 7]);
    }
    return str;
}
```

Checking if Password entered is correct or wrong

Test Case passes if Password matches with initial Password and fails if Password does not matches with initial Password

No	Input	Output	Test Case
1	keshav123	Invalid Password	Fail

Since Password keshav123 is weak, these are following suggestions given to make Password strong :-

k(eshav12G3

keshav1@A23

kesh!Aav123

ke(shav12V3

keGshav1$23

kesXhav@123

keAshav12$3

kesKhav@123

kes$hav12N3

$kesRhav123

No	Input	Output	Test Case
1	keGshav1$23	Valid Password	Pass

Following changes does not improve space or time complexity of the project during execution. But these changes definitely will make user easy to make his/her password stronger.