**Introduction:**

1. **REST Structure :**

As defined by wikipedia, A RESTful web service (also called a RESTful web API) is a simple web service implemented using HTTP and the principles of REST. It is a collection of resources, with four defined aspects:

the base URL for the web service, such as <http://example.com/resources/>

the Internet media type of the data supported by the web service. This is often JSON, XML or YAML but can be any other valid Internet media type. the set of operations supported by the web service using HTTP methods (e.g., **POST, GET, PUT or DELETE**).

The API must be hypertext driven

REST (Representational state transfer) architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

1. **CRUD Structure:**

In implementation REST can be viewed as a CRUD (Create, Read, Update and Delete) designed architectures. The resources are identified by a URL to describe what you would like to perform an operation on, and the request type describes the kind of operation.

|  |  |  |
| --- | --- | --- |
| Operation | SQL | HTTP/REST |
| Create | INSERT | POST |
| Read (Retrieve) | SELECT | GET |
| Update | UPDATE | PUT |
| Delete (Destroy) | DELETE | DELETE |

These actions depend on the resource that has been identified and we have defined 5 standard cases of interaction.

**Software used :**

* Netbeans
* WAMP Server
* Postman extension on Chrome browser
* Windows 7 and higher version

**Language used : php**

**CODE:**

* **index.php:**

<?php

require\_once '../include/DbHandler.php';

require\_once '../include/PassHash.php';

require '.././libs/Slim/Slim.php';

\Slim\Slim::registerAutoloader();

$app = new \Slim\Slim();

//A hello world welcome message

$app->get('/welcome', function() {

$response = array();

$response["Hello World Message"]= "Welcome to the bank ";

echoRespnse(200,$response);

});

//Register a new card

$app->post('/register', function() use ($app) {

// check for required params

verifyRequiredParams(array('number', 'name', 'pin'));

$response = array();

// reading post params

$number = $app->request->post('number');

$name= $app->request->post('name');

$pin = $app->request->post('pin');

$db = new DbHandler();//as soon as you create this file it will connect to ur databasse automatically

$res = $db->register($number, $name, $pin);

if ($res == "CARD\_CREATED\_SUCCESSFULLY") {

$response["error"] = false;

$response["message"] = "Card successfully registered";

} else if ($res == "CARD\_CREATE\_FAILED") {

$response["error"] = true;

$response["message"] = "Oops! An error occurred while registering";

} else if ($res == "CARD\_ALREADY\_EXISTS") {

$response["error"] = true;

$response["message"] = "Sorry, this card already exists";

}

// echo json response

echoRespnse(201, $response);

});

//Get your account balance

$app->post('/balance', function() use ($app) {

verifyRequiredParams(array('number', 'pin'));

$response = array();

// reading post params

$number = $app->request->post('number');

$pin = $app->request->post('pin');

$db = new DbHandler();//data base object

$result = $db->checkBalance($number, $pin);

if ($result == NULL){

$response["error"] = true;

$response["name"] = "Oops! Invalid Credentials";

} else{

$detail = $result->fetch\_assoc();

$response["error"] = false;

$response["name"] = $detail["name"];

$response["balance"] = $detail["balance"];

}

echoRespnse(200, $response);

});

//Credit Transaction route

$app->post('/credit', function() use ($app) {

// check for required params

verifyRequiredParams(array('number', 'pin', 'amount'));

$response = array();

// reading post params

$number = $app->request->post('number');

$pin = $app->request->post('pin');

$amount = $app->request->post('amount');

$db = new DbHandler();

$res = $db->creditTransaction($number, $pin, $amount);

if ($res == "SUCCESS") {

$response["error"] = false;

$response["message"] = "Transaction Successful";

} else if ($res == "INVALID\_CREDENTIALS") {

$response["error"] = true;

$response["message"] = "Sorry, invalid credentials";

}

// echo json response

echoRespnse(201, $response);

});

//Debit Transaction route

$app->post('/debit', function() use ($app) {

// check for required params

verifyRequiredParams(array('number', 'pin', 'amount'));

$response = array();

// reading post params

$number = $app->request->post('number');

$pin = $app->request->post('pin');

$amount = $app->request->post('amount');

$db = new DbHandler();

$res = $db->debitTransaction($number, $pin, $amount);

if ($res == "SUCCESS") {

$response["error"] = false;

$response["message"] = "Transaction Successful";

} else if ($res == "INSUFFICIENT\_FUNDS") {

$response["error"] = true;

$response["message"] = "Oops! You have insufficient funds in your account.";

} else if ($res == "INCORRECT\_DETAILS") {

$response["error"] = true;

$response["message"] = "Sorry, invalid credentials";

}

// echo json response

echoRespnse(201, $response);

});

/\*\*

\* Verifying required params posted or not

\*/

function verifyRequiredParams($required\_fields) {

$error = false;

$error\_fields = "";

$request\_params = array();

$request\_params = $\_REQUEST;

// Handling PUT request params

if ($\_SERVER['REQUEST\_METHOD'] == 'PUT') {

$app = \Slim\Slim::getInstance();

parse\_str($app->request()->getBody(), $request\_params);

}

foreach ($required\_fields as $field) {

if (!isset($request\_params[$field]) || strlen(trim($request\_params[$field])) <= 0) {

$error = true;

$error\_fields .= $field . ', ';

}

}

if ($error) {

// Required field(s) are missing or empty

// echo error json and stop the app

$response = array();

$app = \Slim\Slim::getInstance();

$response["error"] = true;

$response["message"] = 'Required field(s) ' . substr($error\_fields, 0, -2) . ' is missing or empty';

echoRespnse(400, $response);

$app->stop();

}

}

function echoRespnse($status\_code, $response) {

$app = \Slim\Slim::getInstance();

// Http response code

$app->status($status\_code);

// setting response content type to json

$app->contentType('application/json');

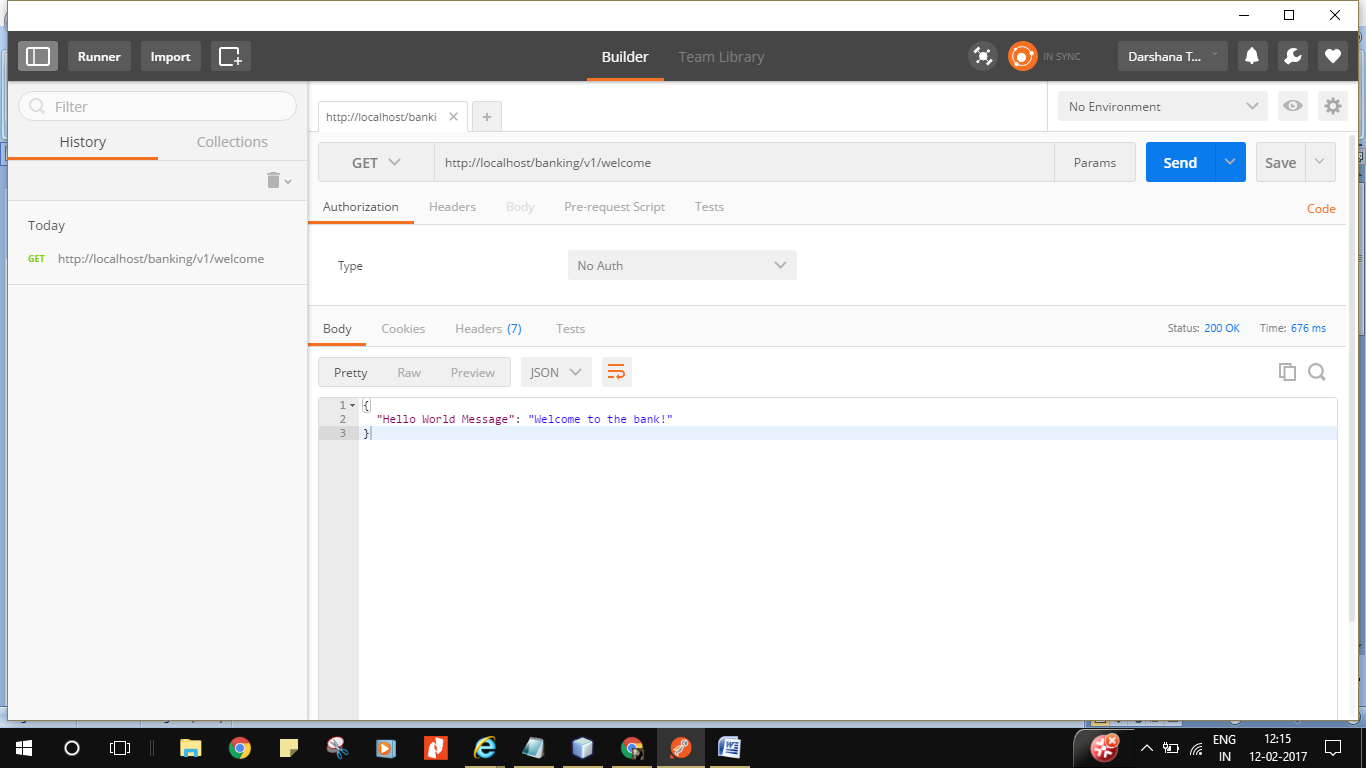
echo json\_encode($response);

}

$app->run();

?>

**Hello World program :**



DBhandler.php:

<?php

class DbHandler {

//Establish Database Connection

private $conn;

function \_\_construct() {

require\_once dirname(\_\_FILE\_\_) . '/DbConnect.php';

// opening db connection

$db = new DbConnect();

$this->conn = $db->connect();

}

// Register a new card

public function register($number, $name, $pin){

require\_once 'PassHash.php';

// First check if card already existed in db

if (!$this->isCardExists($number)) {

$pin\_hash = PassHash::hash($pin);

// insert query

$stmt = $this->conn->prepare("INSERT INTO card(number, name, pin\_hash, status) values(?, ?, ?, 1)");

$stmt->bind\_param("sss", $number, $name, $pin\_hash);

$result = $stmt->execute();

$stmt->close();

// Check for successful insertion

if ($result) {

// User successfully inserted

return $response="CARD\_CREATED\_SUCCESSFULLY";

} else {

// Failed to create user

return $response="CARD\_CREATE\_FAILED";

}

} else {

// User with same email already existed in the db

return $response="CARD\_ALREADY\_EXISTS";

}

}

//Get balance details

public function checkBalance($number, $pin){

if ($this->checkDetails($number, $pin)){

$stmt = $this->conn->prepare("SELECT \* FROM card WHERE number = ?");

$stmt->bind\_param("i", $number);

$stmt->execute();

$result = $stmt->get\_result();

$stmt->close();

return $result;

} else {

return NULL;

}

}

//Credit Transactions - Add amounts to the card

public function creditTransaction($number, $pin, $amount){

if ($this->checkDetails($number, $pin)){

$stmt = $this->conn->prepare("UPDATE card SET balance = balance + ? WHERE number = ?");

$stmt->bind\_param("ii", $amount,$number);

$stmt->execute();

$stmt->close();

$stmt = $this->conn->prepare("INSERT INTO transaction (number, amount, type) VALUES (?,?,'CREDIT')");

$stmt->bind\_param("ii", $number,$amount);

$stmt->execute();

$stmt->close();

return $response="SUCCESS";

} else {

return $response = "INVALID\_CREDENTIALS";

}

}

//Debit Transactions - Deduct amount from the card

public function debitTransaction($number, $pin, $amount){

if ($this->checkDetails($number, $pin)){

$stmt = $this->conn->prepare("SELECT balance FROM card WHERE number = ?");

$stmt->bind\_param("i", $number);

$stmt->execute();

$stmt->bind\_result($balance);

$stmt->fetch();

$stmt->close();

if($balance>=$amount){

$stmt = $this->conn->prepare("UPDATE card SET balance = balance - ? WHERE number = ?");

$stmt->bind\_param("ii", $amount,$number);

$stmt->execute();

$stmt->close();

$stmt = $this->conn->prepare("INSERT INTO transaction (number, amount, type) VALUES (?,?,'DEBIT')");

$stmt->bind\_param("ii", $number,$amount);

$stmt->execute();

$stmt->close();

return $response="SUCCESS";

}

else{

return $response="INSUFFICIENT\_FUNDS";

}

}

else{

return $response="INCORRECT\_DETAILS";

}

}

//Check if the card with same number exists

private function isCardExists($number) {

$stmt = $this->conn->prepare("SELECT number FROM card WHERE number = ?");

$stmt->bind\_param("s", $number);

$stmt->execute();

$stmt->store\_result();

$num\_rows = $stmt->num\_rows;

$stmt->close();

return $num\_rows > 0;

}

//Check the card credentials

private function checkDetails($number, $pin) {

// fetching password hash by number

$stmt = $this->conn->prepare("SELECT pin\_hash FROM card WHERE number = ?");

$stmt->bind\_param("s", $number);

$stmt->execute();

$stmt->bind\_result($pin\_hash);

$stmt->store\_result();

if ($stmt->num\_rows > 0) {

// Found user with the number

// Now verify the pin

$stmt->fetch();//to get the details ie result of sql querry\*

$stmt->close();

if (PassHash::check\_password($pin\_hash, $pin)) {

// User password is correct

return TRUE;

} else {

// user password is incorrect

return FALSE;

}

} else {

$stmt->close();

// user not existed with the email

return FALSE;

}

}

}

**Config.php:**

<?php

/\*\*

\* Database configuration

\*/

define('DB\_USERNAME', 'root');

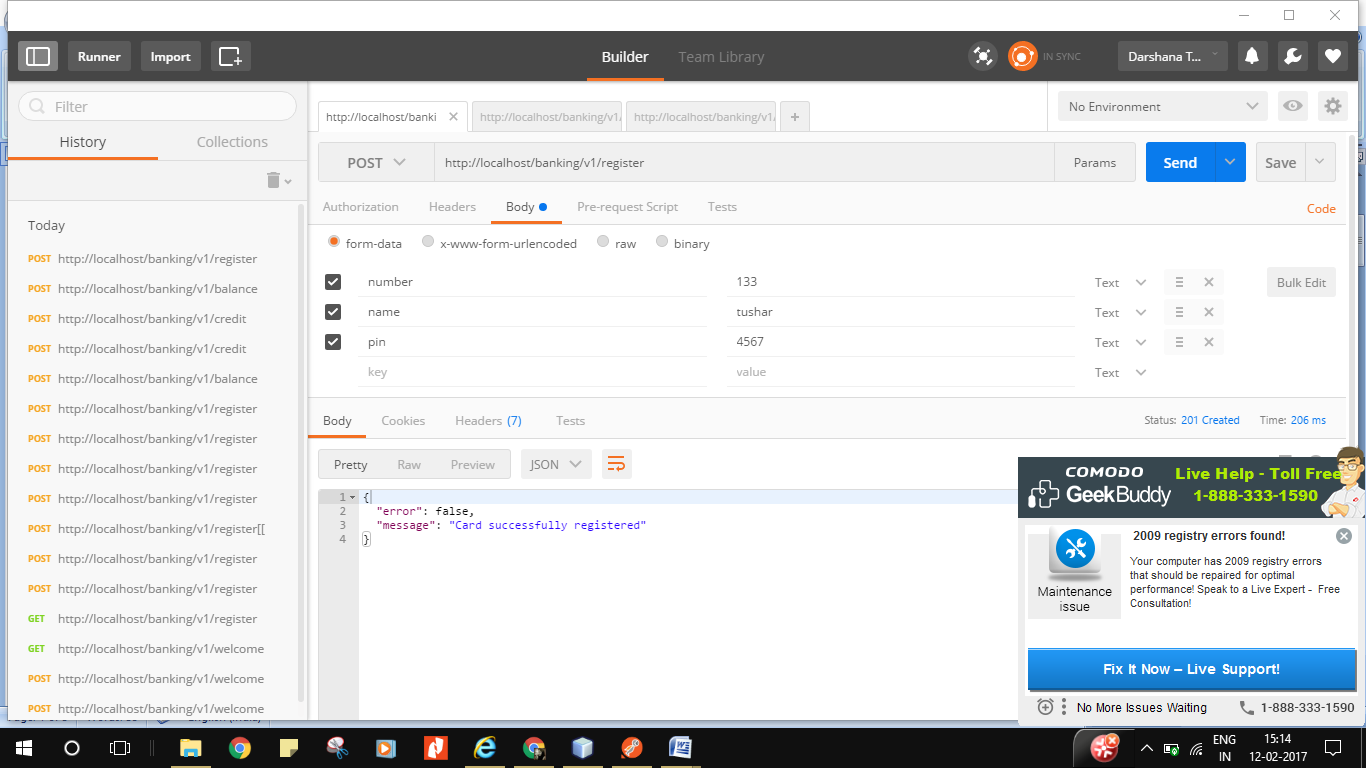
define('DB\_PASSWORD', '');

define('DB\_HOST', 'localhost');

define('DB\_NAME', 'my\_bank');

?>

**Registering new card** :



DBconnect.php:

<?php

class DbConnect {

private $conn;

function \_\_construct() {

}

/\*\*

\* Establishing database connection

\* @return database connection handler

\*/

function connect() {

include\_once dirname(\_\_FILE\_\_) . '/Config.php';

// Connecting to mysql database

$this->conn = new mysqli(DB\_HOST, DB\_USERNAME, DB\_PASSWORD, DB\_NAME);

// Check for database connection error

if (mysqli\_connect\_errno()) {

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

// returing connection resource

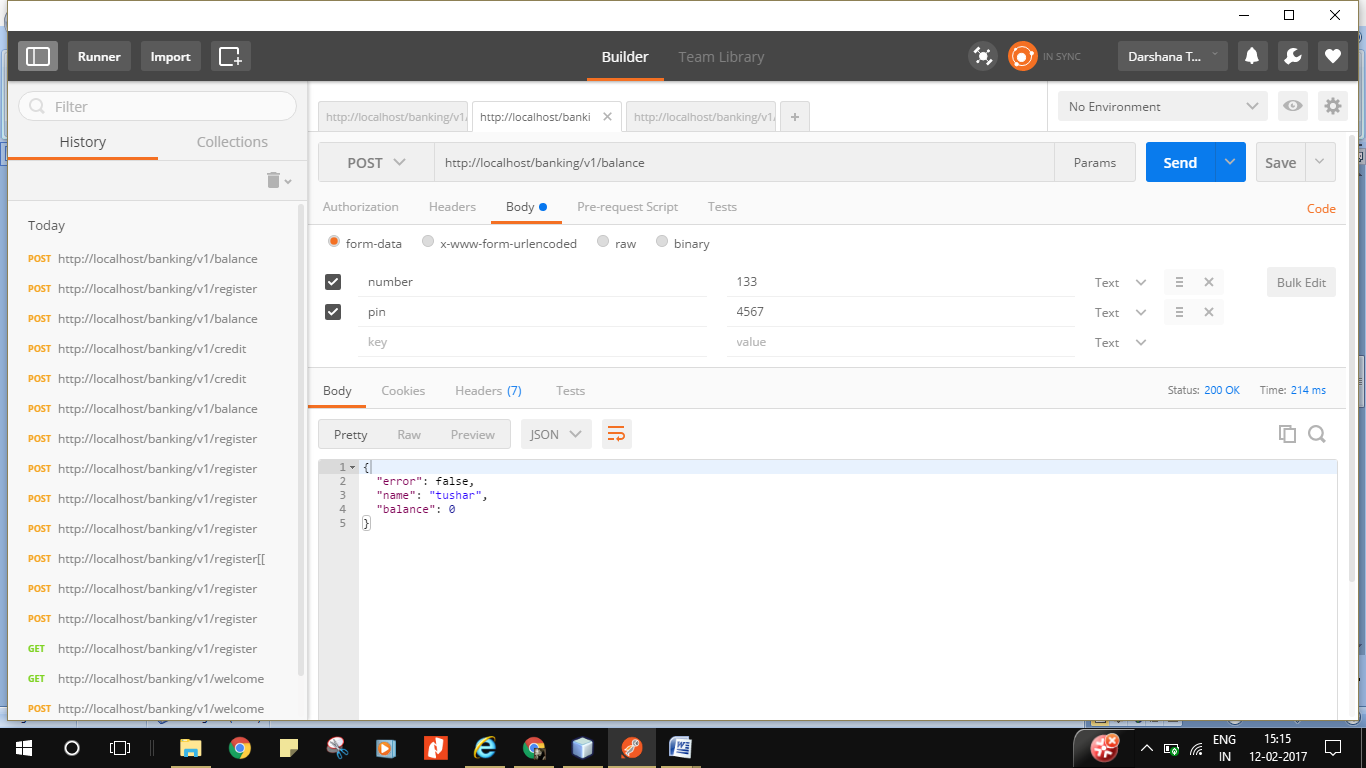
return $this->conn;

}

}

?>

**Checking balance :**



passhash.php:

<?php

class PassHash {

// blowfish

private static $algo = '$2a';

// cost parameter

private static $cost = '$10';

// mainly for internal use

public static function unique\_salt() {

return substr(sha1(mt\_rand()), 0, 22);

}

// this will be used to generate a hash

public static function hash($password) {

return crypt($password, self::$algo .

self::$cost .

'$' . self::unique\_salt());

}

// this will be used to compare a password against a hash

public static function check\_password($hash, $password) {

$full\_salt = substr($hash, 0, 29);

$new\_hash = crypt($password, $full\_salt);

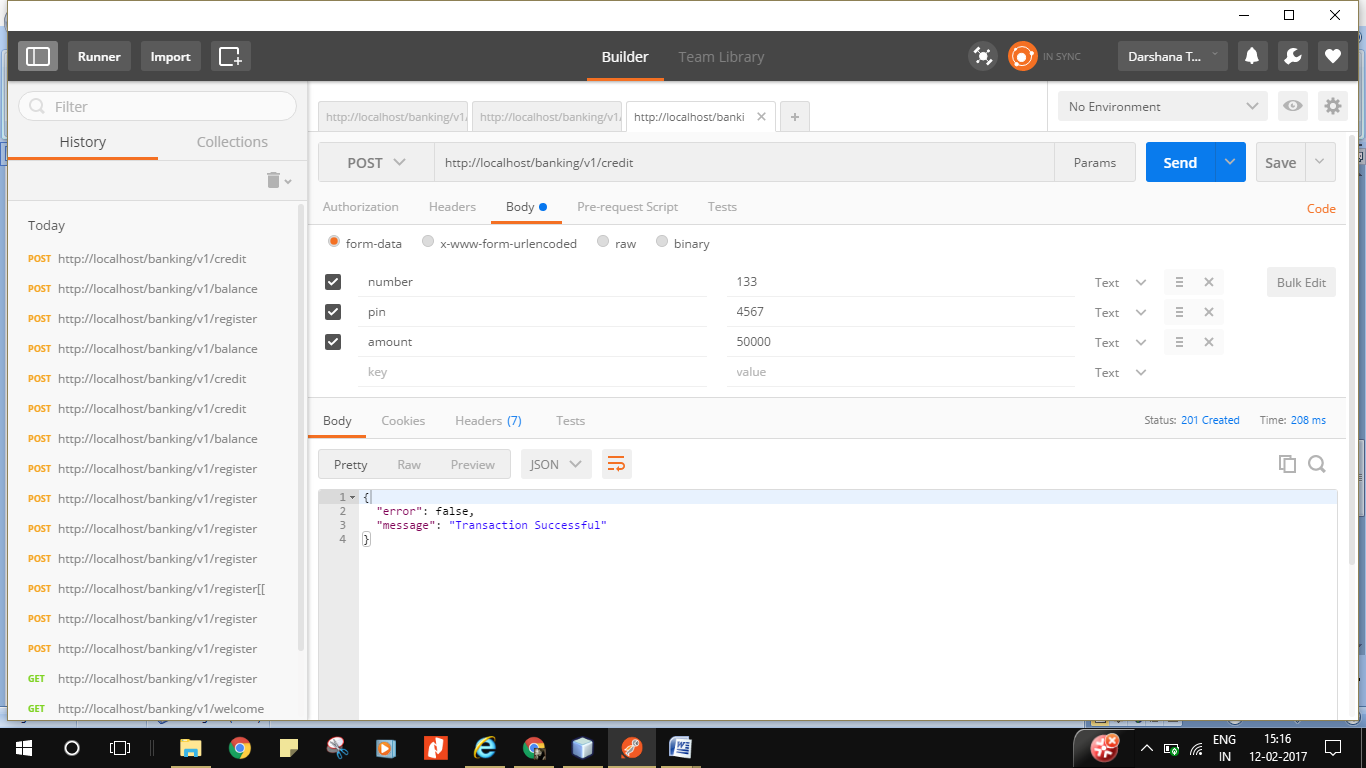
return ($hash == $new\_hash);

}

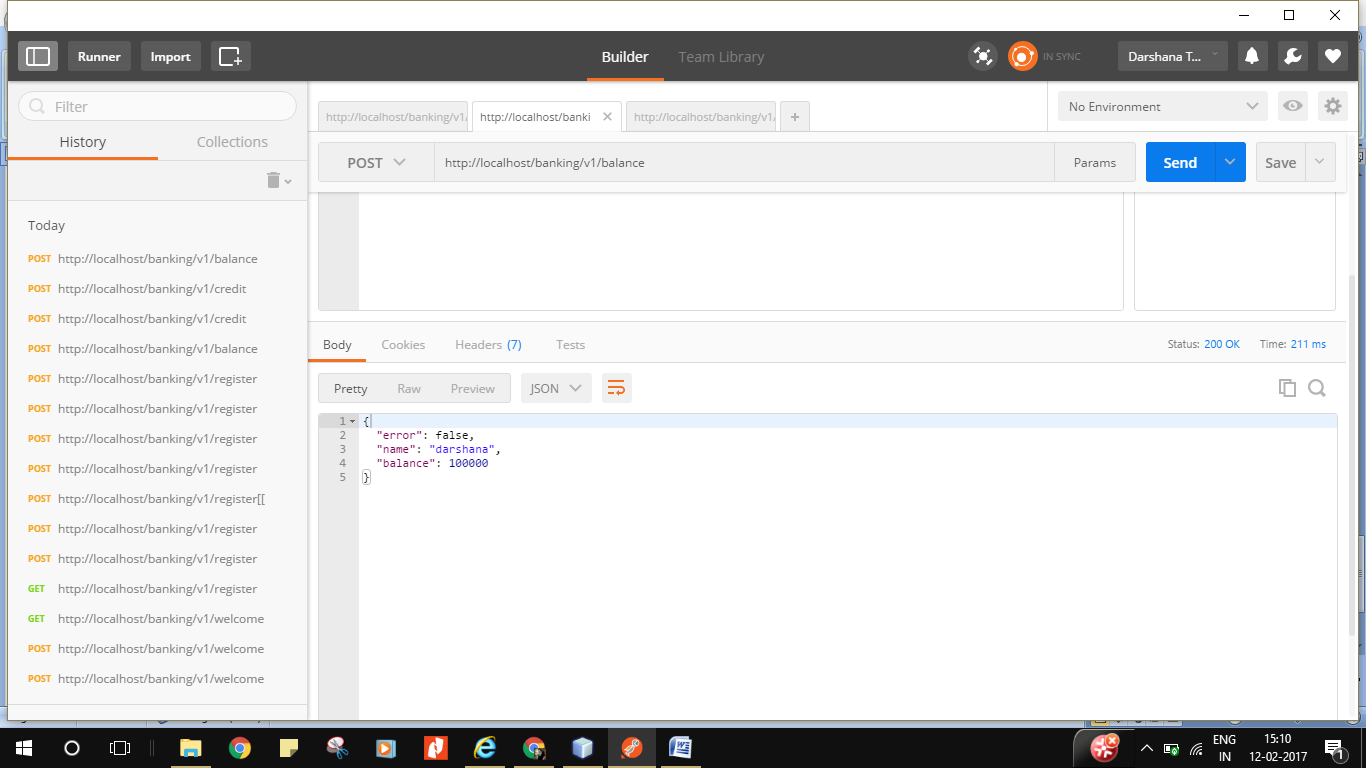
}

?>

**Crediting the amount to the account :**

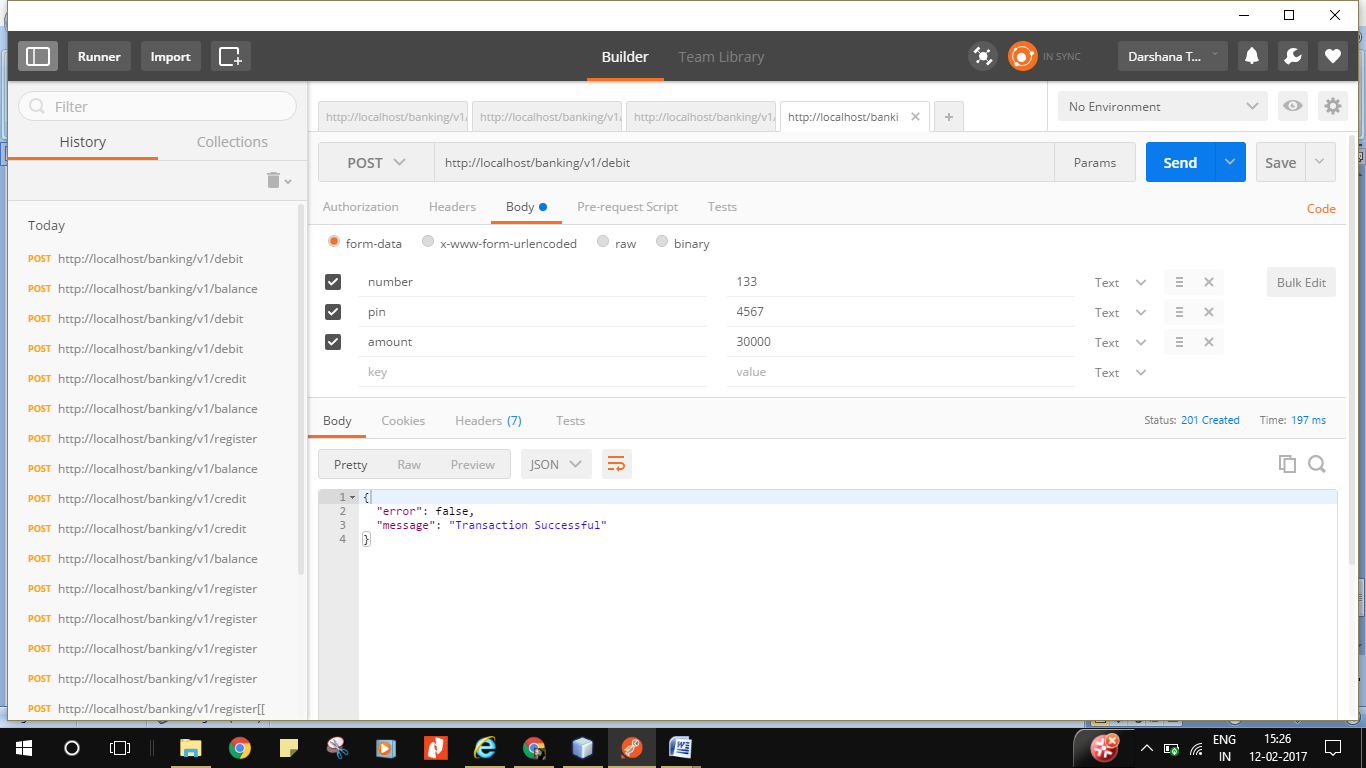


**the fig below: this is updated in your balance / account**

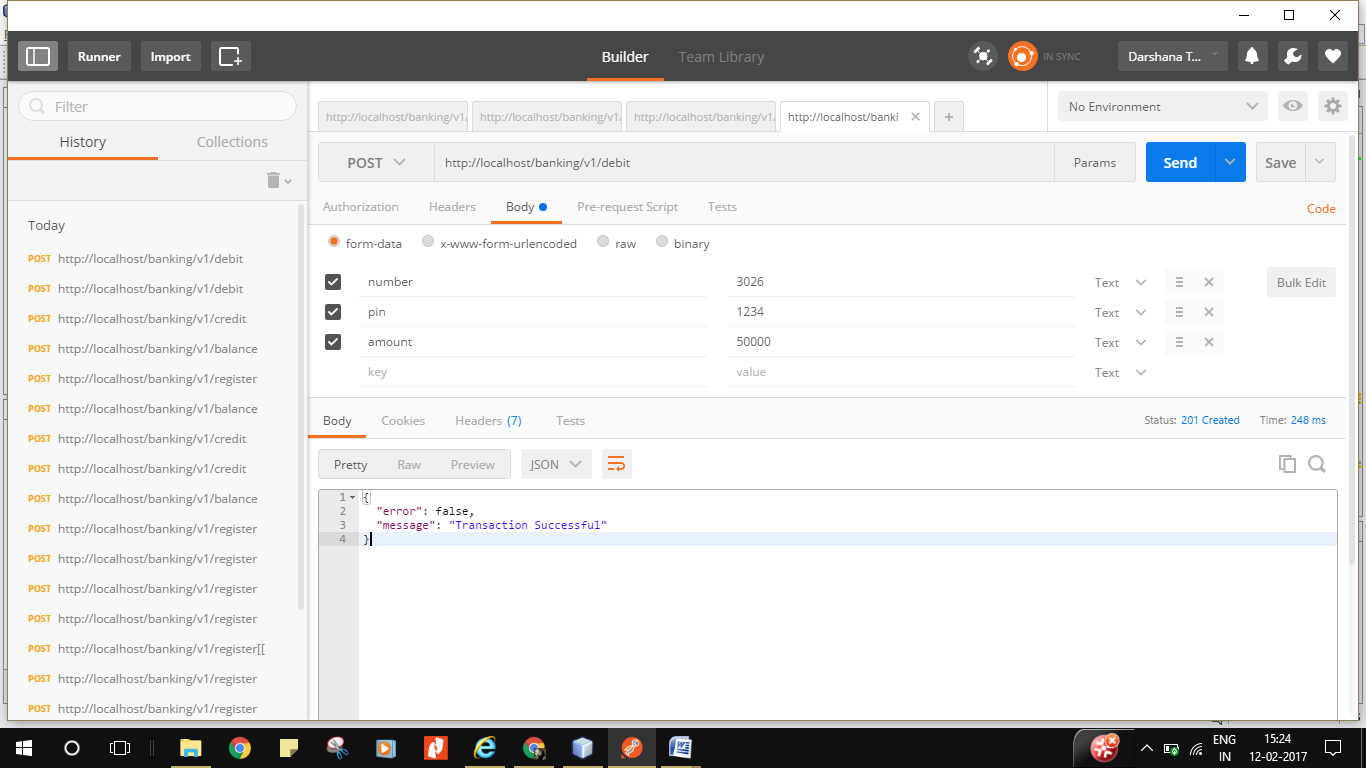


**Debit the amount to the account :**

* if insufficient amount-->



* if amount Sufficient --->



* **Updated balance after deducting amount ----** >

