TEST PAPER – Node Js

The purpose of this evaluation test is to assess the candidate’s broad knowledge and

written communication skills. We recommend the candidate to make the best effort to

fulfill the questionnaire, even though the candidate is not expected to answer each and

Please answer in as clear as possible handwriting. If you use any assumptions when answering

any of these questions, please write them as a part of the answer.

If you write code as part of your answer(s), please write it in Javascript

Question 1: General node.js programming

A.What makes programming in Node.js different than programming in Python, Java or Javascript?

Why?

:- There are two important things that make Node.js different from other server-side languages. First, Node.js uses asynchronous architecture that JavaScript handle well. The server reacts to events and send events/messages to e.g. the database, which is faster. Second, use of JavaScript. By using the same language on the server-side means that the developer can use his JavaScript knowledge both on server and client, and use the same functions if needed.

**Question 2:**

A.Which kind of programs you would not write using Node.js?

CPU heavy applications should not be written in Node.js

B.Why?

Node.js specializes in performing and scaling well for low-CPU, highly I/O bound operations.

C.What would you use instead?

.NET can be used for CPU intensive operations.

Question 3: You are working with a very unstable 3rd party node module which you don’t have access to the source code. The function has the following signature:

3rdparty.unstableFunction(callBackWithTimeout(aCallback, 10000));

function callBackWithTimeout(callback, timeout){

var run, timer;

run = function(){

if(timer){

clearTimeout(timer);

timer = null;

callback.apply(this);

}

}

timer = setTimeout(run, timeout, ‘timeout’);

return run;

}

function aCallback(err, data){

}

Unstable function can sometimes work fine, sometimes return an error but sometimes it can die unexpectedly without invoking the callback. How would you write some code that if after 10 seconds 3rdparty.unstableFunction has not invoked the callback, the callback will be invoked with a Timeout error?

Question4: You have the following app.js file:

----------------------------------------------------------------

var greet= require(‘./greeter’);

greet.greet();

greet.greeting = “Hello new person”;

var greet2=require(./greeter’);

greet2.greet();

----------------

And in the same directory you have a file named greeter.js with the following content: -----------------------

function Greetr(){

this.greeting = “Hello friend!”;

this.greet = function{

console.log(this.greeting);

}

}

module.exports=Greetr;

-----------------------------------------------------------------

a. What will be the printout after running: node app.js ? (greet() and greet2() ) b. Why? c. Can you fix this?

First, the above will result in an error. The this.greet() function will not be accessible in app.js. If the exports are written properly, then the greet instance will print “Hello friend!”, after that the greeting property will be overwritten with “Hello new person”, and greet2() will print the new value “Hello new person”. We can make the code more modularized to print custom messages by keeping the greeting variable private like this:

app.js:

var greet = require('./greeter');

greet.greet();

var greet2 = require('./greeter');

var newGreeting="Hello new person";

greet2.greet(newGreeting);

Greetr.js

var exports = module.exports = {};

greeting = "Hello friend!";

exports.greet = function(hello){

if(!hello){

console.log(greeting);

}else{

console.log(hello);

}

}

Question 5: One of the main issues with Node.js is that there are some tasks that must be done in series one after the other. This creates a “cascading” callback flow that may look like this:

func1(param, callback1(){

… logic ….

func1\_1(params, callback1\_1(){

...logic..

func 1\_1\_1(params, callback1\_1\_1(){

...Logic… };

… more func\_1\_1 logic …

};

} a. How would you eliminate the “nesting hell” b. When calling several different function in parallel, how can you synchronize so you will continue execution only when all of the parallel functions finish their work

:- We can define the callback functions beforehand and then call them inside each functions. However, I would rather use the promise library Bluebird like this:

var Promise = require(‘bluebird’);

var func1 = new Promise(callback1(){

….logic

});

func1.then(func1\_1(){

..logic

}).then(func1\_1\_1(){

});

Question 6: Using MongoDB and Node.js. You have a Mongo DB database with 1 collection named StudentGrades In this collection you have:

{ studentName : “” , // string

gradeType: “” , //can be Exam or Exercise

grade: ‘’, //Integer Value from 0..100

dateOfGrading: “” //Date }

Write a program that takes all the exercise grades for a student, removes the lowest exercise grade and returns a collection with the student name and the average grade for the rest of the exercises. The collection should contain {Name : ‘’ , avgGrade : “”} for all the students