# **HOORIA MUSLIH UD DIN**

# **L16-4197**

# **WEB DEVELOPMENT**

# **RESEARCH ASSIGNMENT**

# **Question 1:**

# **MongoDB:**

MongoDB is an Open Source, NoSQL database management system. MongoDB applications consist of the three basic components:

* Establish a connection to a MongoDB instance
* Logic to access and manipulate database data
* Close the connection to the MongoDB instance

# **Mongoose:**

Mongoose is an Object Document Mapper (ODM) that makes using MongoDB easier by translating documents in a MongoDB database to objects in the program.

# **Code example of insert and create in mongoDB:**

db.collection.insertMany() Inserts multiple documents into a collection.

db.collection.insertMany(

[ <document 1> , <document 2>, ... ],

{

writeConcern: <document>,

ordered: <boolean>

}

)

For create db.collection.createIndex**(keys, options)**

# **Code example of retrieve in mongoDB:**

db.collection.find**(query, projection)**

The projection parameter determines which fields are returned in the matching documents. The projection parameter takes a document of the following form:

{ field1: <value>, field2: <value> ... }

# **Code example of update in mongoDB:**

db.collection.update**(query, update, options)**

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>,

collation: <document>,

arrayFilters: [ <filterdocument1>, ... ]

}

)

# **Code example of Delete in mongoDB:**

# **For deleting one**

db.collection.deleteOne**()**

**db.collection.deleteOne(**

**<filter>,**

**{**

**writeConcern: <document>,**

**collation: <document>**

**}**

**)**

# **For deleting many:**

db.collection.deleteMany**()**

db.collection.deleteMany(

<filter>,

{

writeConcern: <document>,

collation: <document>

}

)

# **Question 2:**

# **Post:**

Post method is used to create. The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line

# **Put:**

Put is used to create or update. The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity should be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI.

Think of them as: PUT = insert or update; POST = insert. So when you make two PUT - you get the one new record, when you do two POSTs - you get two new records. That’s why post was used to update employee data in the Employees sample. If we replace post with put then the server have to make an extra effort to fulfill that request.

# **Question 3:**

# **PATCH:**

The PATCH method requests that a set of changes described in the request entity be applied to the resource identified by the Request- URI.

For partial updates e.g. in case of updating your name In an online university application form some generic method should be used to that there is no danger of losing the data and this method is PATCH. Since it’s a generic format, you can write server- and client-side code once, and share it among a number of applications. Furthermore, your API will become less complex, because it has less URI conventions, leading to more flexibility and making it easier to approach for new developers. Using this approach will also make caches operate more correctly; since modifications to a resource will “travel” through its URL, rather than some other one, the right stored responses will get invalidated. PATCH is atomic, which means that if you need to do some complex changes to the state of a resource, you can do it with confidence. It also means that you can create synthetic resources and PATCH them if you need to organize and arrange changes to the state of several resources.

# **Question 4:**

# **React vs AngularJS:**

React is a JavaScript library. It is not a framework. It is not a complete solution and we will often need to use more libraries with React to form any solution. React does not assume anything about the other parts in any solution. It focuses on just one thing and on doing that thing very well.

We describe User Interfaces with React and tell it what we want (not how to do it). React will take care of the "how" and translate our declarative descriptions (which we write in the React language) to actual User Interfaces in the browser. React shares this simple declarative power with HTML itself, but with React we get to be declarative for HTML interfaces that represent dynamic data, not just static data.

React is much faster and more memory efficient than AngularJS.

AngularJS was born from the first era of web application development. To wit, it was an era focused on the DOM. Coders wrote HTML and CSS then attached functionality to it with JavaScript. JQuery is nothing more than a DOM manipulation library. The DOM was paramount. React rejects the DOM as a thing. Instead, all that exists is the JavaScript application — the state of the application that outputs the DOM. The DOM is a result, not a thing. That is a massive paradigm shift. Suddenly, web pages are genuine applications.

# **Question 5:**

# **Vue.js:**

Vue is a progressive framework for building user interfaces. Unlike other rigid and slow to change frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.

# **Vue In Comparison To React.js and Angular.js:**

Vuejs is just simple and does everything that angularJS and reactJS does, but in a better and strong way. VueJS is not very opinionated, so you can write your own architecture easily of how you want your application to work… however, Essentially, AngularJS is to programmers, ReactJS is to developers, and VueJS is to Architects.

# **Question 6:**

# **Angularjs:**

* The first version of Angular was released in the year of 2010. Some people call this as AngularJS and some people call as Angular 1. But it is officially named as AngularJS.
* AngularJS is completely based on controllers and the View communicates using $scope
* AngularJS doesn't support for mobile devices.

# **Angular 2:**

* The Angular 2 was released in the year of 2016. The important point to note is that Angular 2 is not a simple upgrade of AngularJS. Angular 2 is completely rewritten from scratch and as a result, the ways we write in AngularJS and Angular 2 are completely different.
* Angular 2 is completely a component-based approach. In Angular 2, both, controller and $scope are completely gone. Angular 2 is completely based on the component. Components are the building blocks of an Angular 2 application. The advantage of the Component-based approach is that it facilitates greater code reuse.
* As compared to AngularJS, in Angular 2, the communication among components is very easy. The components can be reused anywhere in the application without much effort.
* In Angular 2, from the unit testing standpoint, the use of components makes the Angular 2 application more testable with minimal effort.
* Angular 2 has more language choices (TypeScript, JavaScript, Dart, PureScript and Elm etc.).
* Angular 2 support mobile services.
* From a performance standpoint, Angular 2 is 5 times faster compared to AngularJS.

# **Angular 4:**

* Angular 4 is released in the year 2017.
* Angular 2and Angular 4 will use the same concept and patterns. Angular 4 simply is the next version of Angular 2.
* Angular 4 is an inheritance from Angular 2. Angular 4 is simply is the next version of Angular 2with few changes
* Changing from Angular 2 to Angular 4 and even future versions of Angular won't be like changing from AngularJS and it won't be like a complete rewrite. It will simply be a change in core libraries.

# **Angular 5:**

* The Angular 5 is released on 1st November 2017.
* Angular 5 has Built Optimizer which helps to removed unnecessary code from your application.
* Angular 5 has Angular Universal State Transfer API and DOM Support which can now share the state of the application between the server side and client side very easily.
* Angular 5 has Compiler Improvements which is one of the very nice features of Angular 5, which improved the support of incremental compilation of an application.
* In earlier versions of Angular, unnecessary new lines, tabs and white spaces were created during the build. Now, in Angular 5, the decision is in your hands whether you need them or not. Angular 5 supports to restrict them (newlines, tabs, and white spaces) in both, the application level or you can restrict them individual component level where you wish to restrict.

# **Question 7:**

# **Linting:**

Linting is the process of running a program that will analyze code for potential errors.

Lint was the name originally given to a particular program that flagged some suspicious and non-portable constructs (likely to be bugs) in C language source code. The term is now applied generically to tools that flag suspicious usage in software written in any computer language.

# **JSLINT:**

JSLint is a tool that can help you to write more reliable and consistent JavaScript code. The tool works by checking your code for a number of common errors. If you’ve ever spent half an hour trying to debug your code only to find that you missed a comma somewhere, you’ll understand how useful JSHint can be.

As well as catching errors in your code, JSLint can be used to enforce coding conventions and style guides. This is really useful if you work on a team as it helps to keep your codebase consistent and easily readable.

# **ESLINT:**

ESLint is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs. In many ways, it is similar to JSLint and JSHint with a few exceptions:

* ESLint uses Espree for JavaScript parsing.
* ESLint uses an AST to evaluate patterns in code.
* ESLint is completely pluggable, every single rule is a plugin and you can add more at runtime.

# **Question 8:**

# **Angularjs example:**

<html ng-app>

<head>

<meta charset="utf-8">

<title>Angular.js Example</title>

<script src="//cdnjs.cloudflare.com/ajax/libs/angular.js/1.2.1/angular.min.js"></script>

</head>

<body>

First name:<input ng-model="firstName" type="text"/>

<br>

Last name:<input ng-model="lastName" type="text"/>

<br>

Hello {{firstName}} {{lastName}}

</body>

</html>

In this example first name and last name is being asked from the user.

# **Ajax:**

<!DOCTYPE html>

<html>

<body>

<h2>The XMLHttpRequest Object</h2>

<h2>Retrieve data from XML file</h2>

<p><b>Status:</b> <span id="A1"></span></p>

<p><b>Status text:</b> <span id="A2"></span></p>

<p><b>Response:</b> <span id="A3"></span></p>

<button onclick="loadDoc('note.xml')">Get XML data</button>

<script>

function loadDoc(url) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this. Status == 200) {

document.getElementById('A1').innerHTML = this. Status;

document.getElementById('A2').innerHTML = this.statusText;

document.getElementById('A3').innerHTML = this.responseText;

}

};

xhttp.open("GET", url, true);

xhttp.send();

}

</script>

</body>

</html>

This is the code to load an Xml file with ajax.