**Ahmad Bilal**

**L164237**

**Web Programming**

**SEC-A**

**Q1-**

Mongodb and Mongoose are two completely different things

Mongodb is the database itself and MongoDB is the npm package, while Mongoose is an object modeling tool for Mongodb.

**Mongodb**:

1. It's a database.
2. This component is governed by the Affero General Public License (AGPL) license.
3. If you link this component along with your proprietary code then you have to release your entire source code in the public domain, because of it's viral effect like (GPL, LGPL etc)
4. If you are hosting your application over the cloud, the (2) will apply and also you have to release your installation information to the end users.

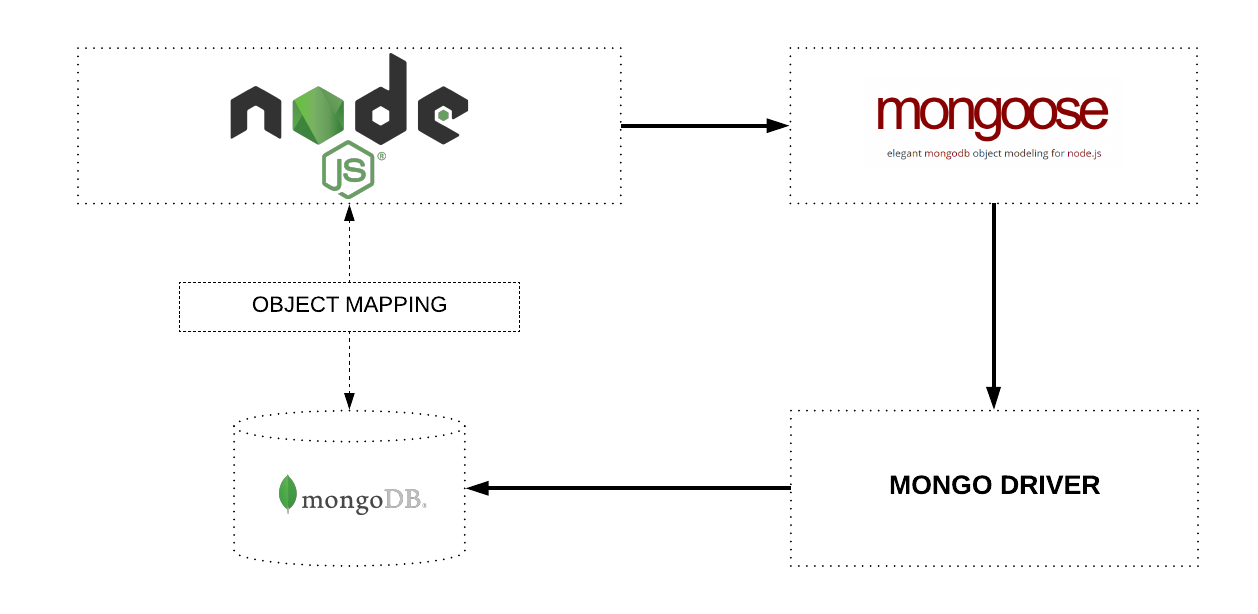
**Mongodb**: native driver in Node.js to interact with MongoDB.

**Mongoose**:

1. It's an object modeling tool.
2. This component is governed by the MIT license.
3. Allowed to use this component along with the proprietary code, without any restrictions.
4. Shipping your application using any media or host is allowed.

**Mongoose** : object data modeling (ODM) library that provides a rigorous modeling environment for your data. Used to interact with MongoDB, it makes life easier by providing convenience in managing data.

Mongoose gives an amazing measure of usefulness around making and working with schemas. Mongoose as of now contains eight SchemaTypes that a property is spared as when it is held on to MongoDB. Mongoose contains a wide range of capacities that enable you to approve, save, delete, and query your information utilizing regular MongoDB capacities



Create or insert operations add new [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) to a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections). If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

* [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne)
* [db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany)

In MongoDB, insert operations target a single [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection). All write operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on the level of a single [document](https://docs.mongodb.com/manual/core/document/).



Read Operations

Read operations retrieves [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) from a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections); i.e. queries a collection for documents. MongoDB provides the following methods to read documents from a collection:

* [db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)



You can specify [query filters or criteria](https://docs.mongodb.com/manual/tutorial/query-documents/#read-operations-query-argument) that identify the documents to return.

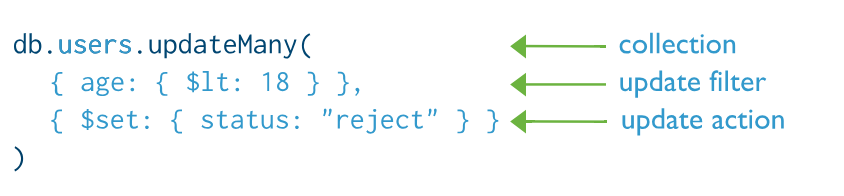
Update Operations

Update operations modify existing [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) in a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections). MongoDB provides the following methods to update documents of a collection:

* [db.collection.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne)
* [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany)
* [db.collection.replaceOne()](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne)

In MongoDB, update operations target a single collection. All write operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on the level of a single document.

You can specify criteria, or filters, that identify the documents to update. These [filters](https://docs.mongodb.com/manual/core/document/#document-query-filter) use the same syntax as read operations.



Delete Operations

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

* [db.collection.deleteOne()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteOne/#db.collection.deleteOne)
* [db.collection.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany)

In MongoDB, delete operations target a single [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection). All write operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on the level of a single document.

You can specify criteria, or filters, that identify the documents to remove. These [filters](https://docs.mongodb.com/manual/core/document/#document-query-filter) use the same syntax as read operations.



**Q2-**

**PUT VS POST:**

|  |  |
| --- | --- |
| PUT method requests for the enclosed entity be stored under the supplied [Request-URI](https://restfulapi.net/resource-naming/). If the Request-URI refers to an already existing resource – an update operation will happen, otherwise create operation should happen if Request-URI is a valid resource URI (assuming client is allowed to determine resource identifier).  PUT /questions/{question-id} | he 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. It essentially means that POSTrequest-URI should be of a collection URI.  POST /questions |
| PUT method is [idempotent](https://restfulapi.net/idempotent-rest-apis/). So if you send retry a request multiple times, that should be equivalent to single request  modification. | POST is NOT idempotent. So if you retry the request N times, you will end up having N resources with N different URIs created on server. |
| Use PUT when you want to modify a singular resource which is already a part of resources collection. PUT replaces the resource in its entirety. Use PATCH if request updates part of the resource. | Use POST when you want to add a child resource under resources collection. |
| PUT is idempotent, so you can cache the response. | Responses to this method are not [cacheable](https://restfulapi.net/caching/), unless the response includes appropriate Cache-Control or Expires header fields. However, the 303 (See Other) response can be used to direct the user agent to retrieve a cacheable resource. |
| Generally, in practice, always use PUT for UPDATE operations. | Always use POST for CREATE operations. |

POST isn't idempotent. So in the event that you retry the demand N times, you will wind up having N assets with N distinctive URIs made on server.

PUT technique is idempotent. So in the event that you send retry a demand on different occasions, that ought to be comparable to single demand alteration.

In the AJAX application we could utilize put ask for as well yet as you can utilize post to refresh an asset however not utilizing indistinguishable URL from the asset you're refreshing. In this way, if the URL to use with PUT/PATCH is/programming interface/vehicles/merchants/1 you'd have/programming interface/autos/dealerupdate to send your POST asks for with body as in your PATCH ask.

Q3-

**When a client needs to replace an existing Resource entirely, they can use PUT. When they’re doing a partial update, they can use HTTP PATCH.**

For instance, when updating a single field of the Resource, sending the complete Resource representation might be cumbersome and utilizes a lot of unnecessary bandwidth. In such cases, the semantics of PATCH make a lot more sense.

Another important aspect to consider here is **idempotence; PUT is idempotent; PATCH can be, but isn’t required to**. And, so – depending on the semantics of the operation we’re implementing, we can also choose one or the other based on this characteristic.

**PUT and PATCH Logic**

Let’s say we want to implement the REST API for updating a HeavyResource with multiple fields:

|  |  |
| --- | --- |
| 1  2  3  4  5 | public class HeavyResource {      private Integer id;      private String name;      private String address;      // ... |

First, we need to create the endpoint that handles a full update of the resource using PUT:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @PutMapping("/heavyresource/{id}")  public ResponseEntity<?> saveResource(@RequestBody HeavyResource heavyResource,    @PathVariable("id") String id) {      heavyResourceRepository.save(heavyResource, id);      return ResponseEntity.ok("resource saved");  } |

This is a standard endpoint for updating resources.

Now, let’s say that address field will often be updated by the client. In that case, **we don’t want to send the whole**HeavyResource **object with all fields**, but we do want the ability to only update the address field – via the PATCH method.

We can create a HeavyResourceAddressOnly DTO to represent a partial update of the address field:

|  |  |
| --- | --- |
| 1  2  3  4  5 | public class HeavyResourceAddressOnly {      private Integer id;      private String address;        // ...  } |

Next, we can leverage the PATCH method to send a partial update:

the overhead of sending whole HeavyResource.

If we have a large number of these partial update operations, we can also skip the creation of a custom DTO for each out – and only use a map:

Finally, let’s write tests for both HTTP methods. First, we want to test the update of the full resource via PUT method:

|  |  |
| --- | --- |
| 1  2  3  4  5 | mockMvc.perform(put("/heavyresource/1")    .contentType(MediaType.APPLICATION\_JSON\_VALUE)    .content(objectMapper.writeValueAsString(      new HeavyResource(1, "Tom", "Jackson", 12, "heaven street")))    ).andExpect(status().isOk()); |

Execution of a partial update is achieved by using the PATCH method:

|  |  |
| --- | --- |
| 1  2  3  4  5 | mockMvc.perform(patch("/heavyrecource/1")    .contentType(MediaType.APPLICATION\_JSON\_VALUE)    .content(objectMapper.writeValueAsString(      new HeavyResourceAddressOnly(1, "5th avenue")))    ).andExpect(status().isOk()); |

We can also write a test for a more generic approach:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | HashMap<String, Object> updates = new HashMap<>();  updates.put("address", "5th avenue");    mockMvc.perform(patch("/heavyresource/1")      .contentType(MediaType.APPLICATION\_JSON\_VALUE)      .content(objectMapper.writeValueAsString(updates))    ).andExpect(status().isOk());  We implemented a simple Spring REST controller to update a Resource via PUT method  and a partial update using PATCH. |

## Q4-

## AngularJS is a legit framework, whereas ReactJS is a store that deals with vision, thus, comparing these two terms is like comparing cheese to chalk. More so, AngularJs is the top-notch JavaScript frameworks and similar to other software groups it has various unique solutions and designs. And on the other hand, developers of ReactJs add a few stores to ReactJS to run it as a usual framework. However, it is obvious that consequential workflow of this language is unlike Angular so comparison is arguable.

High Scalability-

You can easily gamut Angular. All the credit goes to its simple and unique design as well as a solid CLI. While, React is provable that it is reliable and efficient for better workflow. Hence, this framework is very scalable comparatively. And according to Js researches, ReactJs is hits the list when respondents were asked about their satisfaction levels.

Overall Performance-

If we talk about performance, simple recipients in Angular are no more in the picture because these people are called on each service delivery. Thus, it is advisable to optimize BehaviorSubject from ReactJs as it efficiently assists the idea of businesses.

Confidence dose-

From confidence dose I mean dependency injection, which is the heart of debate as it is clashing React model of serviceable programming and stability. However, some sort of dependency injection is inevitable in data binding atmosphere as it assist in detachment where there is no split data coating architecture. More so, Angular serves DI as well and the greatest benefit of the same is it has the capability to include diverse lifecycle for multiple stores.

Well, some of the React models organize some kind of comprehensive app situation that draw to distinct workings, however, it is favorable to the establishment of bugs while vanishing the global situation.

Simple code length-

React is quite simple and easy-to-learn, but it consumes time while setting up a project in React. On the other hand, Angular is not at all easy in any aspects and its natural intricacy creates turmoil as Angular specifically creates third party syntax and stores.

**Paradigm complexity-**

Angular’s performance is quite susceptible with regards to its goal all because of compare and contrast. Furthermore, in any circumstances, you cannot use big paradigms as it has some pros and cons. The benefits include simple coding and provable, but for the disadvantages, you need to split the objects that you typically use and reconstruct it again. However, React in such scenarios, gives you the authority to elect without any extra performance charge. In the end, the results truly depends whether you are a first-class coder or dreadful.

### ****DOM Usage****

DOM is the Data Object Model of a web app. **Angular** uses the browser's DOM, while **React** uses a virtual DOM which help developers manage an extensive database. By using a virtual DOM, you can change any element very quickly and without needing to render the whole DOM. It drastically changes the performance from good to excellent.

## In a nutshell, both the frameworks provide a vigorous set of tools for scalable, quality and reactive web-based applications. However, whoever loves to code in classic Js, React may be the most favorable choice, and those who are looking for established and refined solution, AngularJs might be the popular one.

## Q5-

**Vue** is a progressive framework for building user interfaces. Unlike other monolithic 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](https://vuejs.org/v2/guide/single-file-components.html) and [supporting libraries](https://github.com/vuejs/awesome-vue#components--libraries).

If you’d like to learn more about Vue before diving in, we [created a video](https://vuejs.org/v2/guide/) walking through the core principles and a sample project.

**Angular** is a popular, fully-fledged JavaScript framework used to build modern web applications. React and VueJS are up-and-coming JavaScript libraries, and are used to build web-interfaces.

In this article, we will contrast AngularJS (v1), Angular 2 and React with VueJS by building a small, single-page sample app.

The article will consider the similarities and differences between Angular, VueJS and React in templating, component definition, directive definition, application design, performance, design flexibility and functionality.

## Angular vs React vs VueJS – Similarities and Differences

We will begin by building a demo application using AngularJS (the first version of Angular), as AngularJS has served as an inspiration in the design of VueJS. AngularJS remains to be one of the most popular JavaScript frameworks for developing modern web-applications. Released almost seven years ago, the framework is supported by Google, and aims to provide developers with all the necessary tools, along with a strict set of patterns for front-end development.

After AngularJS, Angular 2 was released. There was no v3. Since Angular v2 to v4 is a progressive enhancement, a majority of the changes are non-breaking and almost all the Angular 2 principles and differences discussed in this article. Going forward, Angular will not be referred to as Angular 4 or 5 or 6. It is going to be called just “Angular”.

[VueJS](https://www.dotnetcurry.com/javascript/1349/introducing-vue-js-tutorial) is an up-and-coming JavaScript library for developing web-interfaces. Released in 2013, VueJS was inspired by AngularJS (v1), and as such borrows the framework’s templating syntax. For example, and as we will see shortly, the syntax for loops, conditionals, model declarations and interpolation all look very similar to that used by AngularJS.

Vue’s focus is on doing two things well: ease of use and rendering speed.

However being a library, as opposed to a framework, means that VueJS does not offer the same amount of functionality that AngularJS offers. For example, whilst Angular provides things such as a HTTP request service or router, VueJS users will require to rely on third-party code if they wish to avail of such functionalities.

### Angular.js vs Vue.js

As VueJS is gaining in popularity, it is important to understand the differences, advantages and disadvantages between VueJS and AngularJS. Therefore this article sets out to explore both VueJS and AngularJS, by building a sample demo application

The hypothetical SurfPics website will allow surfers to display their various beach and surf videos and photographs. In the first section of this article, we will outline the boilerplate code for this demo application. In the subsequent two sections we will then walk you through the development of the application, using first AngularJS and then VueJS. The similarities between the frameworks should become evident quite quickly.

Once we have established the similarity between the two frameworks, we will contrast VueJS to Angular 2, which deviates slightly from its predecessor (AngularJS). Nonetheless, here too, similarities are apparent.

Last but not least, we will discuss the advantages, disadvantages, differences and similarities between a performant alternative to VueJS and AngularJS.

React was developed by Facebook. Similar to VueJS, its objective is to build performant web-interfaces.

**Q6-**

Below is a comparison of AngularJS to Angular, because Angular includes both version 2 and version 4. We compare architecture, language, expression syntax, mobile support, and routing.

**1. Architecture**

**AngularJS**

The architecture of AngularJS is based on model-view-controller (MVC) design. The model is the central component that expresses the application's behavior and manages its data, logic, and rules. The view generates an output based on the information in the model. The controlleraccepts input, converts it into commands and sends the commands to the model and the view.

Angular

In Angular 2, controllers and $scope were replaced by components and directives. Components are directives with a template. They deal with a view of the application and logic on the page. There are two kinds of directives in Angular 2. These are structural directives that alter the layout of the DOM by removing and replacing its elements, and attributive directives that change the behavior or appearance of a DOM element.

In Angular 4, the structural derivatives ngIf and ngFor have been improved, and you can use if/else design syntax in your templates.

Language

AngularJS

AngularJS is written in JavaScript.

**Angular**

Angular uses Microsoft’s TypeScript language, which is a superset of ECMAScript 6 (ES6). This has the combined advantages of the TypeScript features, like type declarations, and the benefits of ES6, like iterators and lambdas.

Angular 4 is compatible with the most recent versions of TypeScript that have powerful type checking and object-oriented features.

Expression Syntax

AngularJS

To bind an image/property or an event with AngularJS, you have to remember the right ngdirective.

To bind an image/property or an event with AngularJS, you have to remember the right ngdirective.

**Angular**

Angular focuses on “( )” for event binding and “[ ]” for property binding.

**Routing**

AngularJS uses $routeprovider.when() to configure routing while Angular uses @RouteConfig{(…)}.

## Advantages and Disadvantages

Because they are Google products, all Angular versions are trustworthy and enjoy great support from Google engineers and the large community of Angular users and developers. However, each version has its own advantages and disadvantages.

**1. AngularJS**

**Advantages**

* It is unit testing ready.
* It has great MVC data binding makes app development fast.
* Using HTML as a declarative language makes it very intuitive.
* It is a comprehensive solution for rapid front-end development since it does not need any other frameworks or plugins.
* AngularJS apps can run on every significant program and advanced cells including iOS and Android-based phones and tablets.

**Disadvantages**

* It is big and complicated due to the multiple ways of doing the same thing.
* Implementations scale poorly.
* If a user of an AngularJS application disables JavaScript, nothing but the basic page is visible.
* There’s a lagging UI if there are more than 200 watchers.

2. Angular 2

**Advantages**

* TypeScript allows code optimization using the OOPS concept.
* It is mobile-oriented.
* It has improved dependency injection and modularity.
* It provides more choice for languages such as Dart, TypeScript, ES5, and ES6 for writing codes.

**Disadvantages**

* It is more complicated to set up compared to AngularJS.
* It’s inefficient if you only need to create simple, small web apps.

## Q7-

## Lint was the name originally given to a particular program that flagged some suspicious and non-portable

## constructs in C language source code. The term is now applied generically to tools that

## flag suspicious usage in software written in any computer language.

## Linting code is already an established part of any (popular) JavaScript project and, in my opinion, has a

## lot of benefits such as:

## 1.Readability

## 2.Pre-code review

## 3.Finding (syntax) errors before execution

JSLint is a static code analysis tool used in software development for checking if JavaScript source code complies with coding rules. It is provided primarily as a web application through jslint.com, but there are also command-line adaptations. It was created in 2002 by Douglas Crockford.  
  
Example:  
(function ($) {

$.fn.loading = function(msg, type, cssClass){

var prefixes = {

warning: 'Warning: ' + msg,

error: 'Error: ' + msg,

info: 'Info: ' + msg,

warning: 'Caution: ' + msg,

};

if (type) {

concatMsg = prefixes[type];

} else {

concatMsg = msg;

}

$(this).each(function() {

var tis = $(this)

if (msg == false) {

tis.html('');

} else {

tis.html(concatMsg);

}

});

}

})(jQuery);

ESLint is an open source, JavaScript linting utility originally created by Nicholas C. Zakas. Code linting is a type of static analysis that is frequently used to find problematic patterns or code that doesn’t adhere to certain style guidelines. There are code linters for most programming languages, and compilers can sometimes incorporate linting into the compilation process.

For further explanation, I will use the following simple script which generates an AST for the given JavaScript code.

var espree = require('espree');

var fs = require('fs');

var code = `let array = [1,2,'b'];

`;

var ast = espree.parse(code, {

ecmaVersion: 6

});

console.log("writing ast to ast.json")

fs.writeFile("ast.json", JSON.stringify(ast, null, 4), function(err) {

if(err) return err;

});

## Q8-

Ajax is the feature which allows you to update the part of the page without update or refresh the page while AngularJS is one of the JavaScript framework (to be specific) client side MVC framework (most says MVW, where W is whatever that is MVC/MVVM )

Even before AngularJS ajax calls were used (today also) to update the particular part of a page and now you can use Angular for too for Ajax calls.One is feature or say way to attain a certain functionality while other is a big framework which include many other functionalities along with Ajax.

You can think of AJAX as the ability to get data from a server without the need to refresh a webpage.However, Angular extends this idea with two-way data binding. So the HTML elements on your front end page are in constant communication with your back end server--and vice versa. Angular can achieve other things as well and is useful for separating your concerns, i.e. separating your data, the functions performed on that data, and how the user sees the data.