**Web Programming Assignment**

**Question-1**

MongoDB vs Mongoose:

MongoDB is a schema-less NoSQL document system which stores data in the form of BSON documents and the structure of these documents can vary as it is not enforced like SQL databases. Mongoose is an **Object Data Modeling (ODM)** library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

In terms of Node.js, MongoDB is the **native driver** for interacting with a MongoDB instance and mongoose is an **Object modeling tool** that makes using MongoDB easier by translating documents in a MongoDB database to objects in the program.

* The Major difference between a Mongoose and a native-MongoDB driver is that, one use Object Relational Mapping (ORM) and another Object Document Mapping (ODM).

Why are we using mongoose package?

* We are using Mongoose package because it manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.
* It provides a lot of convenience in the creation and management of data in MongoDB.

CRUD operations in MongoDB

CREATE:

Create or insert operations add new documents to a collection. 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()
* db.collection.insertMany()

For example:

db.inventory.insertOne({

item: "canvas",

qty: 100,

tags: ["cotton"],

size: { h: 28, w: 35.5, uom: "cm" }

})

READ:

Read operations retrieves documents from a collection; i.e. queries a collection for documents. MongoDB provides the following methods to read documents from a collection:

* dbo.collection.find()

For example:

db.inventory.find({item: "canvas"})

UPDATE:

Update operations modify existing documents in a collection. 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)

For example:

db.inventory.insertOne({

item: "canvas",

},

$set:  
{  
 item: "shoot",   
})

DELETE:

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

* db.collection.deleteOne()
* db.collection.deleteMany()

For example:

db.inventory.deleteOne({

item: "canvas",

})

**Question 2**

Post vs Put:

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.

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.

* Post is used to create document.
* Put is used to create or update document.

Why was POST used to update?

* Because of idempotence nature of PUT, POST was used in the employee AJAX app to update records.
* POST was used to update employee data in the Employees sample AJAX application because in ajax we were using xml and as we know that through xml we can directly make get, post or head request. As two of them are quite similar so we used post in place of put.

**Question 3**

Post vs Patch:

A **PUT** method replaces or changes the whole document rather than just the specific or required field. Thus, a PUT request always contains a full resource.

A **PATCH** request on the other hand, is used to make changes to part of the resource at a location. That is, it PATCHES the resource , changing its properties. It is used to make minor updates to resources and it’s not required to be idempotent.

Can PUT be used for partial updates:

No. In case of updating our name in an online university application form, **Patch** should be used because we are partially updating the information. As it stated above Put is used to change the whole document.

**Question 4**

React vs AngularJS:

The primary difference between AngularJS and ReactJS lies in state management. Angular has data-binding bundled in by default, whereas React is generally augmented by Redux to give unidirectional data flow and work with immutable data.

* **Scalability:**  
  Angular is easy to scale thanks to its design as well as a powerful CLI. React is testable and therefore scalable compared to other frameworks like Vue.
* **Computed Properties:**  
  As far as performance is concerned, plain getters in Angular are out of the scenario because they get called on each render. It is however possible to use BehaviorSubject from RsJS, as it serves the purpose.
* **Dependency Injection:**  
  Dependency injection is a bone of contention as it is contrary to the prevalent React paradigm of functional programming and immutability. But some sort of dependency injection is almost unavoidable in data-binding environments, because it aids in decoupling (and thereby mocking and testing) where there is no separate data-layer architecture. Angular supports DI and one great advantage is the ability to have different lifecycles for different stores.

Some of the common React paradigms deploy some kind of global app state that maps to disparate components, however it is conducive to the introduction of bugs when cleaning the global state on component unmount. On the other hand, a store that is created on component mount and is seamlessly available to the component’s children –is a more useful and often neglected concept. This is out of the box in Angular, but quite easily replicated with MobX as well.

* **Simplicity + Code length**  
  React is quite easy and simple to understand but it takes quite some time to set up a project in React.Angular on the other hand, is not simple by any means. Its inherent complexity sometimes causes confusion and Angular specific 3rd party libraries and syntax.
* **Model Complexity**  
  Angular’s performance is sensitive in terms of scope because of copy-n-compare**.** React however gives you the power of choice without the performance penalty.

**Question 5**

Vue.js:

Vue is a progressive framework for building user interfaces. 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.

Comparison React and AngularJS:

Angular and Vue are frameworks and React is a library to build UI. Vue uses easy javascript and html so it is easier to learn than the other two. AngularJS is used to develop Native apps, hybrid apps, web apps and focus on large-scale, feature-rich applications, React is used to develop SPA and mobile apps and Suitable for modern web development and native-rendered apps for iOS and Android and Vue is used to develop Advanced SPA and started supporting Native apps and is ideal for this purpose. Lastly, Angular is based on MVC model, React and Vue are based on Virtual DOM(Document Object Model).

For example:

new Vue({

el: '#editor',

data: {

input: '# hello'

},

computed: {

compiledMarkdown: function () {

return marked(this.input, { sanitize: true })

}

},

methods: {

update: \_.debounce(function (e) {

this.input = e.target.value

}, 300)

}

})

**Question 6**

* Angularis based on TypeScript whileAngularJS is based on JavaScript. TypeScript is a superset of ES6 and it is backward compatible with ES5. Angular has also benefits of ES6 like: lambda operators, iterators or reflection’s mechanism.
* AngularJS uses terms of scope and controller. To scope a variable you can add many variables that will be visible in View as well as in Controller. AngularJS has also a concept of rootScope. Variables in rootScope are available on all throughout application.
* Angular does not have a concept of scope or controllers. Instead of them it uses a hierarchy of components as its main architectural concept.
* AngularJS has many directives and every developer can also specify custom new directive.
* Angular also has standard directives, but they are used in a bit different way. For example: ng-model in AngularJS means that you want to create two-way binding. If you want to create one-way binding, you should use ng-bind. Angular occurs only ngModel, but if you would write it only in: “[ ]”**,**you’ll get one-way binding. If you want to create two-way binding you must write it in: “[( )]”.
* In Angular, some directives have changed their names like ng-repeat to ngFor.

**Question 7**

Linting:

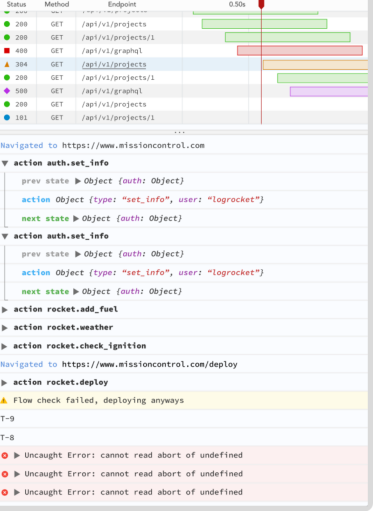
Linting is the process of running a program that will analyze source code to flag programming errors, bugs, stylistic errors, and suspicious constructs. This is most helpful in identifying some common and uncommon mistakes that are made during coding. Now a days, facebook, amazon, twitter etc are using this in their programmes.

Use of JSLint:

Running a Lint program over your source code, helps to ensure that source code is legible, readable, less polluted and easier to maintain.

Use of ESLint:

ESLint is a tool for identifying and reporting on patterns found in 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 is completely pluggable, every single rule is a plugin and you can add more at runtime.



**Question 8**

AngularJS vs AJAX:

**AJAX** as a way to communicate (send requests and get responses) with the server without refreshing. i.e. Asynchronously.

**AngularJS** is a full fledged, front end MVC framework which does a lot more. It extends the above ($http module) with a lot of neat features such as 2-way data binding, templating, filters and directives etc.

* Angular is a complete framework, it can be used for building the whole UI so we can use it wherever we want .
  + Two-Way data binding
  + REST friendly
  + MVC-based Pattern
  + Deep Linking
  + Template
* We can use ajax purposes like sending request to the server or retreating from it.
  + easily manipulate the contents of a webpage
  + apply styles to make UI more attractive
  + easy DOM traversal
  + effects and animation
  + simple to make AJAX calls and