***WEB PROGRAMMING ASSIGNMENT 3***

***RESEARCH ASSIGNMENT***

***L16-4260 CS-A SHAHEER AHMAD***

***Solution1:***

Mongoose is a [MongoDB](https://www.mongodb.org/) object modelling tool designed to work in an asynchronous environment. Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation. Mongoose is higher level and uses the MongoDB driver. It is fairly easy to connect to multiple databases with mongodb native driver while you have to use work arounds in mongoose which still have some drawbacks.

CRUD operations in mongoDB :

Create: 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/).

db.inventory.insertOne(

{ item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }

)

db.inventory.insertMany([

{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },

{ item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },

{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }

])

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

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.

db.inventory.find( {} )

db.inventory.find( { status: "D" } )

db.inventory.find( { status: { $in: [ "A", "D" ] } } )

db.inventory.find( { status: "A", qty: { $lt: 30 } } )

db.bios.findOne(

{

$or: [

{ 'name.first' : /^G/ },

{ birth: { $lt: new Date('01/01/1945') } }

]

}

)

db.bios.findOne(

{ },

{ name: 1, contribs: 1 }

)

Update: 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) *New in version 3.2*
* [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) *New in version 3.2*
* [db.collection.replaceOne()](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne) *New in version 3.2*

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.

db.inventory.updateOne(

{ item: "paper" },

{

$set: { "size.uom": "cm", status: "P" },

$currentDate: { lastModified: true }

}

)

db.inventory.updateMany(

{ "qty": { $lt: 50 } },

{

$set: { "size.uom": "in", status: "P" },

$currentDate: { lastModified: true }

}

)

db.inventory.replaceOne(

{ item: "paper" },

{ item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }

)

Delete: 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) *New in version 3.2*
* [db.collection.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany) *New in version 3.2*

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.

db.inventory.deleteOne( { status: "D" } )

db.inventory.deleteMany({ status : "A" })

***Solution2:***

**PUT:**

PUT puts a file or resource at a specific URI. If there is already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one. PUT responses are not cacheable.

**POST:**

POST sends data to a specific URI and expects the resource at that URI to handle the request. The web server at this point can determine what to do with the data in the context of the specified resource. The POST method is not [idempotent](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.1.2), however POST responses are cacheable so long as the server sets the appropriate Cache-Control and Expires headers.

***Solution 3:***

Put vs Patch:

When a client needs to replace an existing Resource entirely, they can use PUT.

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. it is used for updating the row or document in table completely. In case of updating our name in an online university application form, **Patch** should be used because here we are partially updating the information so using Put for this purpose will be inefficient and won’t work in some cases.

***Solution 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. Some of the other differences are

* **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.
* **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.

***Solution 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:

* 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.
* Angular is based on MVC model, React and Vue are based on Virtual DOM.

Code Example:

Javascript for printing hello in vue:

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)  
  }  
})

<https://vuejsexamples.com/> link for examples of vue.js codes.

***Solution 6:***

Angular IO vs AngularJs:

1. First of all, Angular is based on TypeScript while AngularJS is based on JavaScript.
2. 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. Component is a directive with a template.
3. 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**.
4. Angular has some advantages over AngularJS i.e. modularity, dynamic loading and reactive programming.
5. angular.io. Angular (commonly referred to as "Angular 2+" or "Angular v2 and above") is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations.
6. **AngularJS** is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. **AngularJS's** data binding and dependency injection eliminate much of the code you would otherwise have to write.

***Solution 7:***

Linting:

Linting is the process of running a program that will examine source code to spot 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 programs.

Use of JSLint:

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.

JSLint helps JavaScript programmers by making sure certain coding conventions are followed. JSLint is based on the premise of strict mode, which is available in the fifth edition of the ECMAScript standard. With strict mode, you are making your code run with a set of more restrictive rules than normal.

Use of 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](https://github.com/eslint/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.

***Solution 8:***

Angular is used most in the front end part of the project whereas AJAX is used in the backend for communication with the server.

Angular is a framework. It allows the developer to produce reusable UI components. But, it’s your AJAX that is doing the work below all these layers (in scenarios where you need to communicate with the server from your Angular code).

We can use ajax when we need to send requests to the server e.g. dynamic injection. On the other hand angular is a complete framework, it can be used for building the whole UI so we can use it wherever we want  e.g. automatic completion.