**Phase 3 :**

**Day 1**

**12-02-2022**

**Create scalable and dynamic web site**

Node JS

What is Node JS

Difference between client side and server side scripting language.

Creating basic example using node js

REPL tool

Node JS modules

Core modules

External module

User-defined modules

Fs module (file handling program)

url modules

http module (creating web application using http modules)

express module (external module)(creating web application using express module)

package.json file

web service

SOAP Service

**Rest Full Web Service (REST API)**

Database : no SQL (Mongo DB)

All type of query using mongo db ie basic and adv query.

Connecting mongo db database using mongo db as well as mongoose modules

Creating Web application using MVC style.

CRUD Operation front end Angular and Backed Express JS with Mongo DB database with mongoose module.

Socket IO programming

**Authentication and Authorization.**

**Html, css, JavaScript**

**Html : display the content**

**Css : it is use to display the content in proper format or presentation logic**

**Bootstrap**

**JS : programming on web application or web page or action on web page.**

**DOM**

**jQuery : it is library which provide lot of pre-defined function which help to read, write and update dom properly.**

**Ext JS**

**Coffee JS**

**React JS**

**Angular JS**

**Angular Framework**

**Vue JS**

**D3 JS**

**Backbone js**

**Etc**

**Library or Framework**

**Library is light weighted. Library doesn’t follow standard.**

**Framework is heavy. Framework follow standard rules and regulation.**

**Design pattern : best practise or solution for repeating problem.**

**The implementation of design pattern is taken care by framework.**

**Node JS : Node js is not a library or not a framework. Node js is run time environment for JavaScript program or library or framework.**

**Before Node JS JavaScript only use on client side. Ie Client side scripting language.**

**Front end side**

**Html, css, bootstrap, js and jquery or library or framework.**

**Backend end side**

**Java (JEE ) or Spring boot**

**Asp net**

**Python**

**Php**

**Cgi**

**Node JS : Node JS provided lot of pre-defined modules it may be core module or external module which help to create server side programming language.**

**Using node js (JavaScript program) we can do file handling programing, using js we can create dynamic web page, using js we can create rest api or we can do networking programming or we can connect database.**

**Etc**

**REPL Terminal : Read Eval Print loop**

In node js program we can’t use window as well as document object.

Mean node js doesn’t provide BOM and DOM hierarchy.

Node js provide one of the global object ie

console.

Node JS Modules

Node JS Modules

Module is node js is a simple or complex functionality organized in single or multiple JavaScript files which can re-used through out the application.

3 types of modules

1. Core module
2. Local module or user-defined module
3. External module

Core Modules

Fs (file system)

FS is a pre-defined core modules which provided by node js which help to do file handling synchronously as well as asynchronously.

Syntax to load the modules

let/var referenceName = require(“moduleName”);

JSON is a pre-defined object

Which contains two function which help to convert string to json as well as json to string.

In JavaScript we can create object in three style

1. Function style
2. Literal style : ES5 style
3. Class style : ES6 style

JSON.parse() : it is use to convert string to json.

JSON.stringify(): it is use to convert json or object to string.

Java English Asp.net

Day 2

13-02-2022

Taking the value through keyboards.

readline : readling is a core module which help to take the value through keyboards asynchronously.

process is a pre-defined global object.

Which help to provide processor details.

Node js provided external module it

readline-sync. This module help us to read the value synchronously.

Syntax to install the module

npm install moduleName –g (globally)

npm install moduleName (locally)

npm install readline-sync

http module : node js provide http pre-defined core module which help to create web application using JavaScript Programs.

JEE or spring boot : Java

Asp.net with C#

Php

Python etc

Above technologies provides modules or classes which help to create the web application.

To run those application we need server.

Tomcat

Jboss

Web logic

Apache

IIS

Nginx etc

Non Node JS Server are thread based server.

Program : set of instruction to perform specific task.

Processor : Processor is responsible to execute the task.

Process : time taken to execute the code.

Thread : small execution of code within a process.

Thread also known as light weighted process.

Multi tasking :

1. Process based
2. Thread based

Single processor

Booking Ticket reservation

class Booking {

avl = 1;

}

Booking b = new Booking(); 1st thread

2nd thread

3rd thread

1st client send the request

2nd client send the request

3rd client send the request

Node JS Server found the limitation in thread base server.

Tomcat server

IIS Server

Apache

This sever can handle 100 client request concurrently.

This server can create 100 thread.

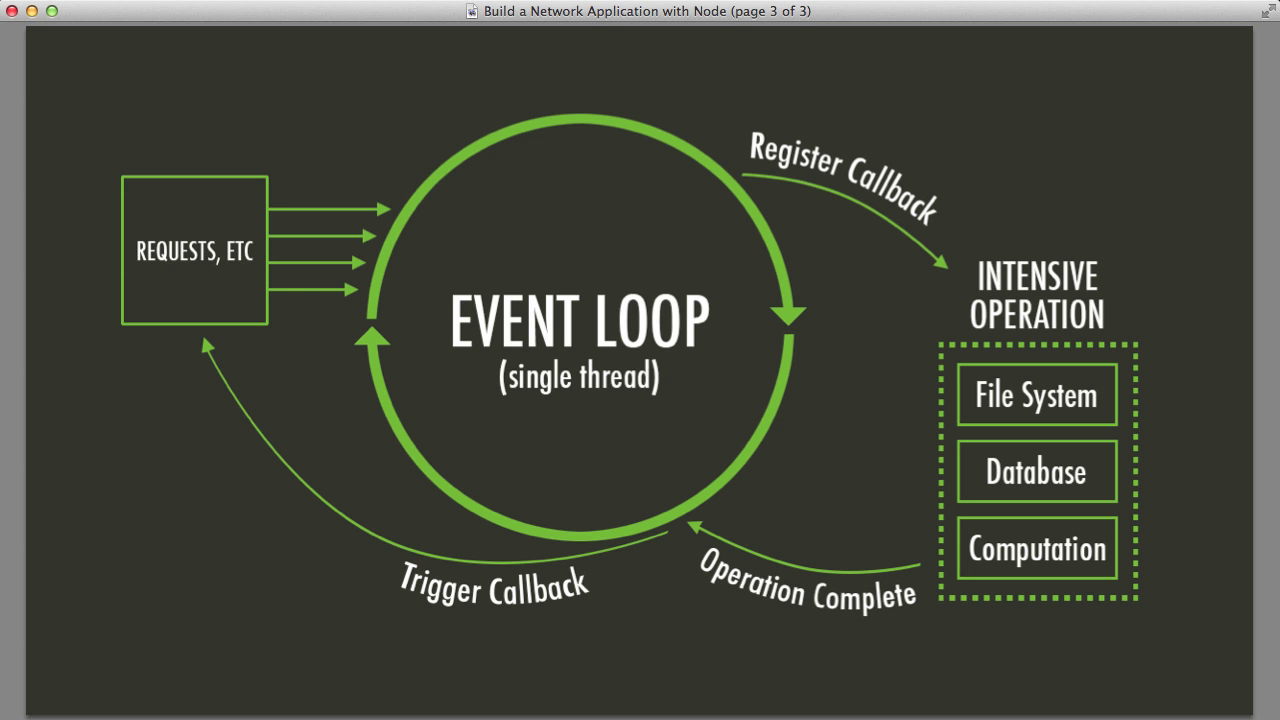
Tomcat Server can handle 100 request concurrently.

101 client send request to server. Thread can be block or lock.

Node JS provide event loop concept.

JavaScript provide great features ie asynchronous and callback.

JavaScript is single thread not multi threading.



Non node JS Server Vs Node JS Server

http module

create http module program folder

node js provide pre-defined it url. This module help use to extract url details.

Node JS provided lot of pre-defined external web framework which help to do develop web application very easily.

Express JS : Express JS is a type of open source web application which help to create web application using node js program.

Express JS internally use http module to take the request from a client.

Express JS support all http protocol method like get, post, put and delete.

Express wrap http module.

Create the folder express module

npm install express locally

or

npm install express –g globally

package.json : This file contains all dependencies or modules configuration details.

npm init : which command is use to create the package.json file

we have to install express module

npm install express

npm install : it check in package.json file what are dependencies required to develop the application and those dependencies install locally.

npm uninstall modulename

npm uninstall mongodb

if method is get data send through url using query param concept using key=value pairs

url?key=value&key=value&key=value

if method is post data send through body part.

By default in Express data is not enable.

To enable this data we have to take the help to external module ie body-parser.

Express JS 3.x version we have to download body-parser module separately.

But from Express 4.x version onwards body-parser module part express JS.

In Express js we can take the help of use function.

To add other middleware modules.

Middleware means between client and server module

app.use(middlewaremodulename);

**Phase 3 :**

**Day 4**

20-02-2022

var num =[10,20,30,40];

var flag=0;

for(var i=0;i<num.length;i++) {

if(num[i]==100) {

flag++;

}

}

if(flag>0) {

console.log(“Present”);

}else {

Console.log(“not present”);

}

let result = num.find(obj=>obj==300);

if record present it will return that value else return undefined.

if(result!=undefined) {

consle.log(“Present”);

}else {

Console.log(“Not present”);

}

Express JS with HTML as a view.

View is a static we can’t do any dynamic task or we can’t use any variable value in html page which is part of node js application.

Express JS provide lot of pre-defined view-engine or express template. Which help to do dynamic task on view side.

Jade

Pug

Etc

Express js provided external module ie express-generator which help to create express js application with any of the view engine.

npm install express-generator –g

create separator folder as

express generator example

now to create the project we have to use the command as

express project-name

express project-name --view pug

express demo-app

cd demo-app

open the project in vs code

first run the command to install the external dependencies require to run the sample express project

npm install

to run the application run the command as

npm start

Create new folder

npm init

express demo-app

If we use any one of the express JS engine like Jade or pug or etc. We can do dynamic task on that engine.

The view engine tightly coupled with backend technologies like Express JS.

Req java request

Client HSBC XML/JSON SBI

Res Java Express JS

Js response

Express JS

Amazon

Payment Paytm Java

Google pay Asp.net

Net banking

Credit Python

Debit card Express JS

Web Service : Giving the service for web application when both application running using different technologies.

2 types of web service

1. SOAP base web service
2. RESTFull web service

SOAP : Simple Object Access Protocol.

Service consumer

Service Provider

Service Broker

WSDL file

UDDI Registry

SOAP Web service is base upon SOA (Service Oriented Architecture).

In SOAP web service we can consume and produce the data only in the form of XML (eXtensible Mark up language).

XML is very heavy.

RestFull Web service

Representational State Transfer

Using RestFull Web Service we can expose our resource as a web service in any format base upon client requirement.

Rest full web service can use only http or https protocol.

Restfull web service is just a style.

Restul web service is architecture style to expose our resources ie Express JS as we Web Service to consume and produce any format of data.

In Server side technologies like

Java Servlet or jsp or spring framework

Asp.net

Php

Express JS

Using this technologies we can receive the request from a client and response to client base upon the request.

ES6 object in JS

class Employee {

constructor(id,name,age) {

this.id=id;

this.name = name;

this.age = age;

}

}

let emp = new Employee(100,”Ravi”,21);

SOAP Web service

XML

<Employee>

<Id>100</Id>

<Name>Ravi</Name>

<Age>21</Age>

</Employee>

SOAP Request

SOPA Response

JSON : Java Script Object Notation

{“id”:100,”name”:”Ravi”,”age”:21}

Java

If we make Express JS application as a REST Full Web Service any other technologies can invoke or all our backend technologies.

View can be

Java

Asp.net

Php

Angular

React JS

Or other REST Client application

RestFull Web service using Express JS module

Using Express We will create REST API.

Those API is use to create, delete, update and retrieve resources.

Generally resources known as entity

Employee

Customer

Product

Order

Login

Etc

Create the folder

Express JS with REST API

Create the package.json file using npm init command

npm install express

REST API

Get method

1. Get the data in text format
2. Get the data in json format
3. Get employee resource in json format
4. Get all employee resources in json format.

Using get method passing the data to express js application

1. Query param
2. Passing single value : URL?key=value
3. Passing multiple value : URL?key=value&key=value

If view is normal html page internally they use query param concept.

1. Path param
2. Passing single value : URL/v1
3. Passing multiple value : URL/v1/v2

If view is normal html then we can use query param concept.

If view technologies is angular or react or command base rest client then you can use path param.

Post method : This method is use to create the resources

Store the information in file or database or array.

Patch method : This method is use to update existing resource partially.

Employee resource

We want to update age using id property

We want to update name using id property

Put method: Put method also update the existing resource.

Patch is use to update one more or more property but put method is use to update all property.

Delete method : This method is use to delete the resources.

We take the property value using path param.

**Phase 3 :**

**Day 5**

26-02-2022

Product Entity

Pid

PName

Price

Image

Create the Folder Express CRUD Operation

Backend : Express JS with REST API with FS module

Create the package.json file using npm init command

npm install express

then create product.json file

[]

Now create app.js file to do all express js operation.

npm install nodemon –g

nodemon is a external module if we run the express js application using nodemon module if we do any changes it automatically refresh the application.

Now onward rather than

node app.js

We have to use

nodemon app.js

Frontend : Angular

In Frontend folder create the angular project

ng new angular-product-app

ng new angular-product-app

routing : no

styling : css

Run these 3 command in angular project

create the component using command as

ng g c product

create the service using command as

ng g s product

create the model class using commands as

ng g class product

CORS : Cross Origin Resource Sharing :

Two domain going to communication each others.

Angular application running on port number 4200

Express JS running on port number 9090

Node Js provide external module ie cors enable to frontend technologies to access their resources.

In backend technologies install cors modules

npm install cors

add as a middleware

27-02-2022

Add Operation

**Template driven form**

Flow wise template to component

Easy to use

Good for simple form

**Model driven form or reactive form**

flow wise component to template

WE need good knowledge of typescript and angular API.

Good for complex form.

FormGroup is a collection of more than one form controller.

FormControl is to bind specific form field ie textfield, passwordfield, radiobutton, checkbox etc.

<input type=”text”/>

**05-03-2022**

**File system :** In file system we store the data permanently.

Limitation of file system is

Data redundancy (mean we can store duplicate record in file system).

Data consistency : format of the file.

Txt, doc, pdf, excel

.json or .xml or .cvs etc.

Security : we can make file read mode or write mode.

Database :

Data : raw fact

Information : processed data or meaningful data.

Database : storing data and information in table format.

DBMS : Database Management System :

DBMS is a software which help to store the data in table format.

Excel : DBMS

MS Access : DBMS :

Table -🡪 Employee

Id Name Salary Attribute (id,name,salary)

1 Ravi 12000 records

2 Ramesh 14000

3 Lokesh 16000

1 Mahesh 18000

RDBMS : Relational Database Management System

**Trainer\_Student\_Info**

Tid TName Tech SId SName Age

1 Raj Java 100 Seeta 21

1 Raj Java 101 Meeta 22

1 Raj Java 102 Leeta 23

Trainer

PK

TID TName Tech

1 Raj Java

2 Ravi Python

Student

PK FK

SId SName Age TSId

100 Seeta 21 1

101 Meeta 22 1

102 Leeta 23 2

103 Veeta 22 2

Trainer\_Student

TSId TID SID

1111 1 101

2222 2 101

All RDBMS database like MySQL, Oracle, Db2 are schema base database.

Employee

Id Name Age city PhNumber

1 Raj 21 null null

2 Ravi 22 null null

3 Ramesh 23 Bangalore null

4 Ajay null Mumbai 9900

JSON : Angular support JSON

Express JS : Support JSON

If we use RDBMS database then we have to convert JSON data into SQL Query format and vice-versa.

Structure : Table format

Semi structure : JSON or XML

Un structure :

Retrieving the records from more than one table is slower than retrieving the records from single table.

No SQL Database

Key-value : redis

Graph database : Neo4j

Document Oriented Database : Mongo DB

Column family : Cassandra DB

Mongo DB Database : Mongo DB is no sql open source high performance database. It is use to store the data using document in the json format.

Plz download mongo db exe file window user.

Mac user

<https://treehouse.github.io/installation-guides/mac/mongo-mac.html>

Before start mongo db server you have to create

data

* db folder inside C drive.

Inside a C driver create data folder and inside that create db folder.

Then open the command prompt inside a below path

**C:\Program Files\MongoDB\Server\5.0\bin**

**Then type mongod command to start mongo db database.**

**Then open another command prompt in same location and write mongo command. Which provide a terminal to do mongo db database query.**

**Then write all query in mongo terminal**

**Cntr + L : to clear the screen.**

**show databases This command is use to display all database present in mongo db.**

**show dbs : This command is use to display all databases present in mongo db.**

**To create a database in mongo db we have to use the command as**

**use databaseName : This command create the database if database not exits if database present it will switch to that database.**

**To check all table present in database we can use the command as**

**show collections;**

**Or**

**show tables;**

**In Mongo DB collection is known as tables.**

**Syntax to create the collection**

**db.createCollection(“Sample”);**

**In mongo db record is known as document.**

**Inside a collection we can store more than one document.**

**In mongo db we can store data in document in the form of json.**

**Syntax to insert the document in collection**

**db.CollectionName.insert({key:value});**

**db.Sample.insert({name:”Ramesh”});**

**db.getName(); This command is use to find the current database name.**

**View the documents from collection**

**db.CollectionName.find();**

**db.Sample.find();**

**Mongo Db by default internally provided pre-defined attribute ie \_id with generate unique id for every document if document contains same type of field with same value or different.**

**If you want to pass \_id value you can do it. But you can’t change the \_id name. so it is just like primary key. But in Mongo DB primary key always \_id**

**Create 🡪Emp**

**\_id,Name,Age,Salary,City**

**Insert 5 to 8 document.**

**Filter the document using index position**

**db.Emp.find()[0];**

**db.Emp.find()[1];**

**retrieve specific index position particular value.**

**db.Emp.find()[0].name;**

**db.Emp.find()[1].city;**

**to retrieve specific fields or more than one field for all documents.**

**db.CollectionName.find({condition},{filterFields});**

**db.Emp.find({},{name:1});**

**db.Emp.find({},{name:1,\_id:0});**

**db.Emp.find({},{name:1,\_id:0,age:1});**

**retrieve more than one field with particular index positon**

**db.Emp.find({},{name:1,\_id:0,age:1})[0]**

**skip the number for document from collection**

**db.Emp.find().skip(2);**

**display number of documents from a collection**

**db.Emp.find().limit(2);**

**sort the documents using fields.**

**db.Emp.find().sort({age:1}); Ascending order as age**

**db.Emp.find().sort({age:-1}); descending order as age**

**db.Emp.find().sort({city:1})**

**db.Emp.find().sort({city:-1})**

**filter the fields from a document**

**db.Emp.find({\_id:1});**

**db.Emp.find({city:"Bangalore"});**

**db.Emp.find({age:22});**

**db.Emp.find({salary:{$gt:20000}});**

**db.Emp.find({salary:{$gte:20000}});**

**db.Emp.find({salary:{$lt:20000}});**

**db.Emp.find({salary:{$lte:20000}});**

**db.Emp.find({salary:{$eq:21000}});**

**db.Emp.find({salary:{$ne:21000}});**

**more than one conditions using $and / $or operator**

**db.Emp.find({$and:[{\_id:1},{name:"Ravi"}]});**

**db.Emp.find({$and:[{\_id:2},{name:"Ravi"}]});**

**db.Emp.find({$or:[{\_id:2},{name:"Ravi"}]});**

**update the query**

**db.Emp.update({\_id:1},{$set:{salary:18000}}) : it update only one document.**

**db.Emp.updateMany({city:"Bangalore"},{$set:{city:"Bangaloru"}}) : it can update more than one documents.**

**Remove query**

**db.Emp.remove({\_id:1});**

**if record present it will update else it will insert.**

**db.Emp.update({\_id:1},{$set:{name:"Ravi",salary:18000}},{upsert:true})**

**remove field for first document**

**db.Emp.update({},{$unset:{salary:1}});**

**remove fields for all documents.**

**db.Emp.updateMany({},{$unset:{salary:1}});**

**Drop the collection**

**db.sample.drop();**

**06-03-2022**

**Creating collection array value in documents.**

**Student**

**SId SName Age Sub1 Sub2 Sub3 Sub4 etc**

**SId SName Age**

**100 Ravi 21**

**101 Ramesh 22**

**Subject**

**SubId SubjectName**

1. **Math**
2. **GK**
3. **Eng**
4. **Phy**

**db.Student.insert({\_id:1,name:"Ravi",age:21,sub:["Math","Phy","Che"]});**

**Mongo DB Relationship**

**In RDBMS we will achieve the relationship using primary key and foreign key.**

**In Mongo DB we can do relationship using two ways**

1. **Embedded style (only single collection)**
2. **Linking style (more than one collection)**

**Employee {\_id:100,name:”Ravi”,age:21}**

**Address : {city:”Bangalore”,state:”Kar”}**

**{\_id:100,name:”Ravi”,age:21,add: {city:”Bangalore”,state:”Kar”}}**

**One employee has one address : one to one relationship**

**Employee {\_id:100,name:”Ravi”,age:21}**

**Project :{pid:1111,tech:”Java”}**

**Project : {pid:222,tech:”Python”}**

**Project : {pid:333,tech:”Angular”}**

**{\_id:100,name:”Ravi”,age:21,projectInfo:[**

**{pid:1111,tech:”Java”},**

**{pid:222,tech:”Python”},**

**{pid:333,tech:”Angular”,location:{lid:11,location:”Mumbai”}}**

**]}**

**One employee working more than one project at the same time**

**One to many relationship**

**In one collection we can achieve one to one and one to many at same time.**

**db.Employee.insert({\_id:1,name:"Ravi",age:21,add:{city:"Bangalore",state:"Kar"}});**

**db.Employee.insert({\_id:2,name:"Ramesh",age:22,add:[{city:"Bangalore",state:"Kar"},{city:"Mumbai",state:"Mh"}]});**

**db.Employee.insert({\_id:4,name:"Mahesh",age:28,project:[{pid:111,tech:"Java"},{pid:222,tech:"Angular"}]});**

**db.Employee.find().pretty()**

**linking style**

**Trainer**

**\_id TName tech**

**100 Ravi Java**

**101 Ramesh Angular**

**102 Raju Python**

**db.Trainer.insert({\_id:100,tname:"Ravi",tech:"Java"});**

**Student1 (We are storing trainer is in Student collection)**

**\_id SName Age TSId**

**1 Seeta 21 100**

**2 Meeta 22 100**

**3 Leeta 23 101**

**4 Keeta 22 102**

**5 Teeta 24 [100,101]**

**db.Student1.insert({\_id:1,sname:"Seeta",age:21,tsid:db.Trainer.find()[0].\_id});**

**db.Student1.insert({\_id:5,sname:"Teeta",age:25,tsid:[db.Trainer.find()[0].\_id,db.Trainer.find()[1].\_id]});**

**Student2**

**\_id SName Age TrainerInfo**

**1 Seeta 21 {tid:100,tname:”Raj”,tech:”Java”}**

**2 Meeta 22 {tid:101,tname:”Ramesh”,tech:”Angular”}**

**3 Leeta 23 {tid:101,tname:”Ramesh”,tech:”Angular”}**

**4 Keeta 22 {tid:102,tname:”Raju”,tech:”Python”}**

**5 Teeta 24 [**

**{tid:100,tname:”Raj”,tech:”Java”},**

**{tid:101,tname:”Ramesh”,tech:”Python”}**

**]**

**db.Student2.insert({\_id:2,sname:"Meeta",age:22,tsid:db.Trainer.find()[0]});**

**db.Student2.insert({\_id:5,sname:"Teeta",age:25,tsid:[db.Trainer.find()[0],db.Trainer.find()[1]]});**

**Two approach**

**In Student1 collection we stored only trainer id if you want to retrieve trainer details using Student1 collection we have to use aggregate operator.**

**But in Student2 we store whole trainer document details. Not required any others operators.**

**Aggregation function :**

**Mongo DB provided aggregation function which help to groups multiple documents and then we can perform mathematical or aggregate operation on field which contains numerical value and return a single or more multiple result depending upon the group.**

**Aggregate function like a group by, having clause clause in RDBMS.**

**Aggregate function with $lookup operator**

**db.Student1.aggregate([{$lookup:{from:"Trainer",localField:"tsid",foreignField:"\_id",as:"TrainerDetails"}}]);**

**db.Trainer.aggregate([{$lookup:{from:"Student1",localField:"\_id",foreignField:"tsid",as:"StudentDetails"}}]);**

**Aggregate with $group, $match, $project by operator**

**$match**

**db.EmployeeDetails.aggregate([{$match:{city:"Bangalore"}}]);**

**db.EmployeeDetails.aggregate([{$match:{deptId:100}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$city",totalSalary:{$sum:"$salary"}}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$city",minSalary:{$min:"$salary"}}}]);**

**Index :**

**Index are very important concept in all database it may be sql or non sql database.**

**Using Index we can improve the performance.**

**Index are special data structure that store a small part of collection or table ‘s data in a way data we can queries or search document or records very easily.**

**Info -🡪 Collection**

**100 document**

**Field city 🡪 Bangalore**

**db.Info.find({city:”Bangalore”});**

**using some syntax if we create index on city field**

**bucket -🡪Bangalore**

**Delhi**

**Mumbia**

**We can create the index on single field or multi field.**

**By default \_id field contains index.**

**db.Employee.getIndexes();**

**using above command we can find the index details.**

**Syntax to create the index**

**db.EmployeeDetails.createIndex({city:1});**

**db.EmployeeDetails.dropIndex({city:1});**

**creating unique index**

1. **classes**

**using node js how to connect mongo db database**

**using module mongo db and mongoose**

**3 class mongodb and mongoose**

**Exprees MVC with Angular**

**1 Authentication and Authorization**

**1 Socket IO**

**Last 1 days Angular Pending Topics**

**12-03-2022**

**Connecting Mongo DB database using Node js**

1. **using mongodb module**
2. **using mongoose module**

**create the folder**

**connecting database using mongodb module**

**open the folder in vs code**

**create package.json file using the command as**

**npm init**

**npm install mongodb**

**run the mongod and mongo command in bin folder of mongodb database.**

**Create the folder product api using mongo db module**

**Create package.json file using command npm init**

**npm install express**

**npm install mongodb**

**controller layer : this layer is responsible to take the request and base upon request give the response client.**

**This layer call repository layer to do the operation on collection.**

**router layer : router concept provided by express js it check sub path and http method which is receive from main file and base upon the path and http method it will call specific controller methods.**

**Repository layer or DAO(Data access object) layer : This layer is responsible to do pure database logic.**

**This layer call getColletion function to connect database and specific collection.**

**And depending upon the requirement it will do the specific operation on that collection.**

**App.jsp --🡪Router -🡪Controller --🡪Repository -🡪config file --🡪Database**

**App.js file**

**Load the express module, link to config module to connect the database, run the application on server, link to router file and pass the request to router file**

**Router.js file**

**It check sub path and http method base upon sub path and method it will pass the request to specific controller method.**

**Controller : this file receive the request from router file and base upon the request it will call specific repository method. Inside this file you have to do specific request and response task.**

**Repository : This file is responsible to write pure database logic.**

**config : this file provide the database connection**

**Mongoose : Mongoose is a external module provided by node js which help to connect the database.**

**Mongooses support ODM (Object Data Modelling).**

**Mongoose internally use mongo db module it provide extra features as Schema. Schema help the create the structure for the collection means name of the field and data type for the field provided by schema.**

**First we have to create the schema using schema we will create the model. Model will help use to do the operation on collection ie insert, delete, update and retrieve.**

**Controller layer**

**Repository layer**

**Config layer**

**Router layer**

**Model layer : In this layer we have to configure schema and model details.**

**Main file**

**Create the folder**

**Customer API using mongoose module**

**Open the project in VS code.**

**Then create the package.json file using npm init command**

**npm install express**

**npm install mongoose**

**please create the folder**

**config**

**router**

**controller**

**repository**

**model**

**app.js**

**19-03-2022**

**Authentication and Authorization**

**These concept we use in every web application develop using any technologies.**

**In Simple Authentication is the process of verifying who a user (who are you) and Authorization is the process of verifying what they have access to in that application or resources (what you are allowed to do).**

**Http is stateless protocol. When any client send the request to server every time sever consider as new client.**

**Session Tracking : Session is a collection of request and response within a particular period of a time.**

**1st req**

**2nd + session Id**

**Client server**

**1st res + cookies (session Id)**

**2nd res + sessionId**

**Cookies: it is a small text file created by server when any client send first request to server. This file contains unique session id. And cookies files store in client machine.**

**Cookies is not a secure any one can capture our data.**

**JWT (JSON Web Token) JWT are a good way of securing transmitting information between two parities or client and server.**

**Today JWT is very popular for handling authentication and authorization.**

**JWT divided into two part**

**Header section**

**This section provide the details about what type of algorithms going to use to generate the token.**

**Body section (payload data).**

**This section hold the information out user data (like user\_id or email\_id etc) which help to generate unique token using unique property.**

**Login emaildId**

**And password**

**Client server**

**JWT**

**Res + JWT (which generate using unique id with respective user).**

[**http://localhost:9090/api/customer/StoreCustomerInfo**](http://localhost:9090/api/customer/StoreCustomerInfo)

**http://localhost:9090/api/customer/**updateCustomerAge

**[http://localhost:9090/api/customer/](http://localhost:9090/api/customer/StoreCustomerInfo)**[deleteCustomerInfo/:\_id](http://localhost:9090/api/customer/StoreCustomerInfo)

**[http://localhost:9090/api/customer/](http://localhost:9090/api/customer/StoreCustomerInfo)**[findAllCustomer](http://localhost:9090/api/customer/StoreCustomerInfo)

**Node Js provided external module ie bcryptjs which help to convert password in decrypt format.**

**Please install**

**npm install bcryptjs**

[**http://localhost:9090/api/user/register**](http://localhost:9090/api/user/register)

**{**

**"email":"admin@gmail.com",**

**"password":"admin",**

**"name":"Admin",**

**"user\_type":"admin"**

**}**

[**http://localhost:9090/api/user/signIn**](http://localhost:9090/api/user/signIn)

**{**

**"email":"admin@gmail.com",**

**"password":"admin"**

**}**

**Once the account details is correct we have to generate the token ie JWT (Json Web Token)**

**Node Js using external module jsonwebtoken which help to generate token.**

**So we have to install jsonwebtoken**

**npm install jsonwebtoken**

**In this application we have to 4 REST API ie create, delete, update and retrieve.**

**To verify token we will add the middleware**