**24-07-2021**

**Day 1**

**Phase 3**

**Node JS**

Node JS Concept.

Basic programming using Node JS

Node JS Modules core

fs module

util module

taking value through keyboards using different external modules.

url module

http module

client and server application

setting the data for node js application using get and post methods.

External module

Express modules

Rest API

Testing rest API get, post, put and delete methods using plugins or cURL.

Database

Mongo DB : No SQL Database

Basic query and adv query

Connecting database through node js using mongo db as well as mongoose modules.

Using Express module creating MVC project with mongoose module to connect the database.

Connecting Angular Application to Express MVC module

MEAN Stack

CRUD Operation

Angular 🡪 Express ->Mongoose 🡪 Mongo DB

Socket programming concept using Node JS

**Node JS :** Node JS is a run time environment for the JavaScript library or JavaScript framework.

Every browser contains JavaScript plugin to check or run the client side JavaScript programs.

In Client Side JavaScript program we can use DOM and BOM.

BOM : Browser object model

DOM :document object model

Client side JavaScript program we have to run on Browser using internal JS file or external JS file.

Front end Team back end team

HTML/CSS/JavaScript JEE-Servlet/JSP/EJB

jQuery and bootstrap Spring boot

Asp.net

Php

CGI

**Using JavaScript creating dynamic web page, receive the request and response from a client application, storing data in file, connecting database, creating rest API etc.**

**Node JS goal is to provide an easy way to build scalable networking programs.**

**Node JS provide non blocking IO Operation through networking environment.**

**Because JavaScript support callback function and asynchronous operation.**

**XMLHttpRequest and ActiveXObject**

**Using API if we send the Data we have wait for the acknowledgement may be success or failure.**

**Node JS is an open source, platform independent runtime environment for JavaScript programs.**

**It provide an event driven architecture and non blocking IO operation.**

**Node JS program are executed using V8 JavaScript engine. Chrome browser use V8 engine.**

**Node JS**

**Node JS is not a framework**

**Node JS is not a good for beginner**

**Web Application or Server or REST API.**

**Not a multi thread**

**Node JS = Runtime environment + JavaScript library /JavaScript framework + core module or external modules.**

**Node JS**

**Running Node JS JavaScript using command prompt**

**To install new version of node JS**

**npm install node**

**then set the path may be temporary or permanent.**

**For temporary set path=URL;.;%PATH%**

**set path=C:\Users\91990\node\_modules\.bin;.;%PATH%**

**Node JS provide its own terminal to run the Node JS Programs.**

**REPL : Read Eval Print Loop**

**In Node JS program we can’t use BOM and DOM.**

**console**

**Node JS provider global object ie console which help to display the output in REPL terminal or normal console.**

**process :**

**Node JS Provide another pre-defined global object ie process. This object is use to find the processor details.**

**Node JS Modules**

**Node JS modules is a simple or complex functionality organized in a single or multiple JavaScript files which can be reused through the Node JS application like a Package in Java technologies.**

**Using Node JS module we can achieve separation of concern.**

**Types of modules**

1. **Core Module**
2. **Local module (user-defined modules)**
3. **Third party modules**

**Core Modules : All node js application by default provide few set of modules which come under the category core modules**

1. **fs : File system**
2. **os : Operating system**
3. **net : networking : socket programming**
4. **url : if we want to extract the data from URL**
5. **http : to create basic web application.**
6. **https : to create secure basic web application.**

**etc**

**Fs Module :**

**Node JS provide pre-defined Core module ie fs module which help to do file handling program may be synchronously as well as asynchronously.**

**Syntax to load the module in ES5 style**

**var/let variableName = require(“moduleName”)**

**var fs = require(“fs”);**

**25-07-2021**

**Day 2**

**Phase 3**

**Reading the value through console in node js**

**readline : readline is external module which help to receive the value asynchronously.**

**Syntax to install the external module**

**Globally**

**npm install –g moduleName**

**or**

**locally**

**npm install moduleName**

**package.json file**

**This file hold the configuration details about node js projects.**

**If we install the module locally the package.json file hold details about that modules.**

**readline-sync : This module help use to read the value synchronously.**

**url module : This module is use to find the URL details like port number, localhost, query params etc.**

let urlInfo = "http://localhost:9090/welcome?name=ravi&age=21";

let url = require("url");

//console.log(urlInfo);

let urlObj = url.parse(urlInfo,true);

console.log(urlObj);

console.log(urlObj.protocol)

console.log(urlObj.hostname)

console.log(urlObj.port)

console.log(urlObj.pathname)

console.log(urlObj.query)   // string consider if parse second parameter is false

let obj = urlObj.query;     // it consider as reference if parser second parameter is true

console.log("Name is "+obj.name)

console.log("Age is "+obj.age)

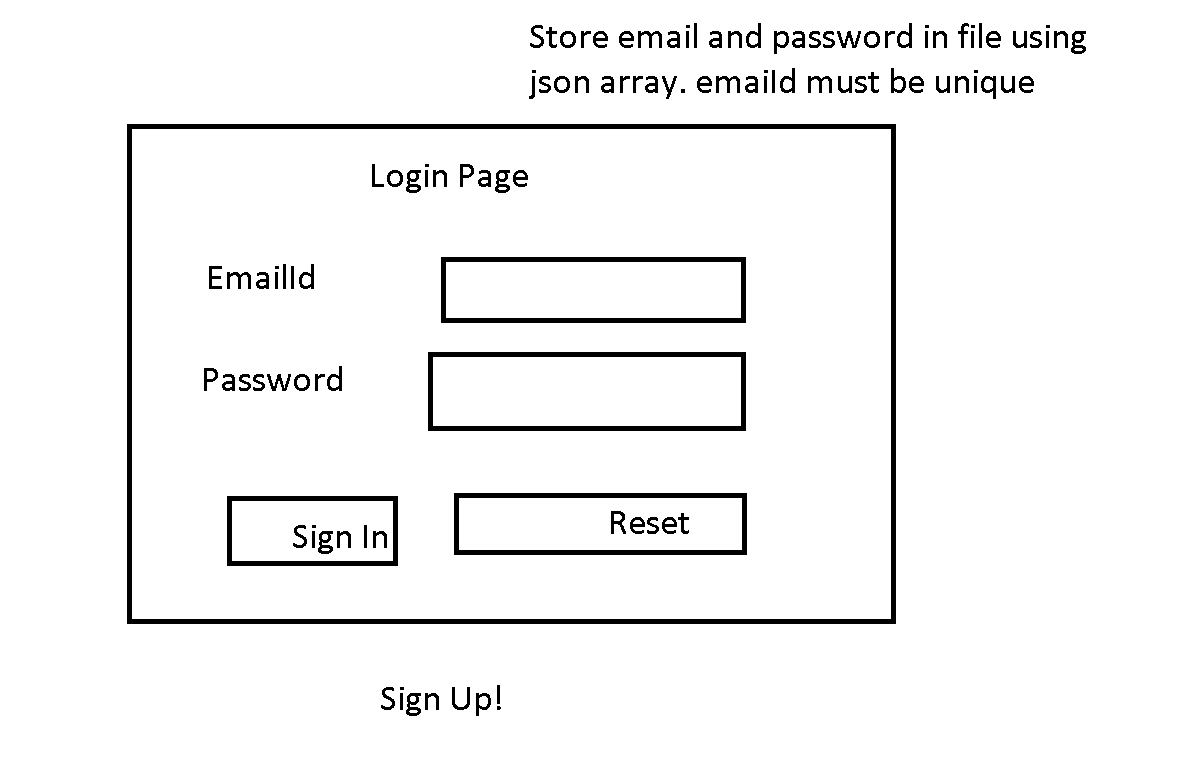
**http module :**

**Node js provide pre-defined module ie http module. Using http module we can create own server using Node JS programs.**

**Jsp, asp, php etc.**

**Tomcat, welogic, jboss, IIS etc**

**http and fs modules**

****

**Difference between Normal Server and Node JS server**

**Non node js server are by default thread base**

**Tomcat, WebLogic, IIS Server etc.**

**Tomcat / IIServer :**

**1,000 : this server can response 1,000 client concurrently.**

**1,001 client if send the request.**

**Online Booking**

**class Booking {**

**int avl=1;**

**}**

**One object created and inside that multiple client equal to multiple thread created.**

**3 client send request.**

**In Thread base server request can be block or lock.**

**Node JS Server provide even driven architecture framework.**

**JavaScript is single thread.**

**We can do or achieve multiple task execute independently using callback and asynchronous communication.**

**31-07-2021**

**Day 3**

**Node JS Express Module**

**http is pre-defined core module with the help of http module we can do core functionality for the web application.**

**Node JS provide External Express module. Express module wrap http module provide more functionality.**

**Like it support get, post, put and delete methods,**

**Using Express module we can create REST API, It support middleware module etc.**

**npm install –g express**

**or**

**npm install express**

**To create package.json file**

**To create package.json**

**npm init**

**It ask some details. Please hit the enter key.(enter continuously). given confirmation yes**

**Or**

**Short cut to create package.json file**

**npm init –y**

**If we install the external module locally then package.json file contains all modules details using dependencies attribute as well as it download the node\_module folder.**

**Like**

**npm install express**

**\_\_dirname : it provide current directory path of our application.**

**If form method is get the information send through URL using URL re-writing technique.**

**Get Method : we can send the request through**

1. **Browser URL**
2. **Hyperlink**
3. **Submit button with method = Get**

**Post method : We can send the request through**

1. **Form submit button with method = post**

[**http://localhost:9090**](http://localhost:9090) **: default message**

[**http://localhost:9090/loginGet**](http://localhost:9090/loginGet) **: open login page with Get method**

**after click submit button of login page with get Method**

[**http://localhost:9090/checkUser?user=Raj&pass=1234**](http://localhost:9090/checkUser?user=Raj&pass=1234) **:**

[**http://localhost:9090/loginPost**](http://localhost:9090/loginPost) **: open login page with Post method**

**after click submit button of login page with post method**

[**http://localhost:9090/checkUser**](http://localhost:9090/checkUser)

**if method is post data send to server through request body part.**

**Node JS provided external module ie body-parser. Which help to enable the body data from request.**

**npm install body-parser**

**then load the module**

**With Express JS html template plugin :**

**Pug**

**Haml.js**

**EJS Plugin**

**Jade template**

**More**

**These above help use to display dynamic value of server side technologies in html page.**

**If we use Express JS template, the view pages become tightly coupled with Express JS or backend technologies.**

**Rest Full Web service came in picture.**

**If we create any application using Java, Asp.net, Python, Node JS with respective view technologies.**

**The application can’t call or re-usability by another application.**

**View**

**JSP Servlet**

**Php Php**

**Asp C# and ASP**

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

1. **SOAP Base Web Service : Simple Object Access Protocol**

**SOAP is base upon the SOA ( Service Oriented Architecture).**

**In SOAP Web service we can consume and produce only XML format data.**

**XML is very heavy**

**WSDL ( Web Service Description Language).**

**This file provide the Web Service details develop in different language.**

**WSDL**

**Provide**

**Service**

**Details**

**In xml format.**

**Java asp.net**

**We have to create Java code with the help of wsdl file**

1. **REST Full Web Service:**

**Representational State Transfer : Exposing server side resource as a web service.**

**JSP/Servlet, Asp.net, Php, Node (Express JS).**

**If we expose our resources as Rest API ie Asp.net, Php, Java(Spring boot), Node Express JS.**

**Any other application can call our REST API.**

**Rest Client can be**

**Java , Asp.net, Php, Node JS(Express), angular, React JS or other REST Client.**

**REST Client is Angular**

**In REST Web service we can call REST API using http protocol method with URL.**

**Create new folder**

**ExpressREST\_API**

**npm init –y (folder name must be only word) : package.json**

**npm install express**

**npm install body-parser**

**get() : string message**

**get() : json message**

**get() : object in json format**

**get() : array object in json format**

**passing the value to get method from REST Client Application.**

1. **Query Params :by default plain html web page form with get methods internally follow query param concept.**

**URL?key=value&key=value&key=value**

1. **Path Params**

**URL/path/value1**

**URL/path/value1/value2/value3**

**Day 4**

**01-08-2021**

**Entity : Customer, Person, Project, Employee, Order**

**Get() : Get Resources: retrieve all customer , person, employee details. Retrieve customer details base upon the id.**

**Select query**

**Post() : Create Resource : insert employee details, customer details, order details, etc.**

**Insert query.**

**Browser Plugin :**

**Postman plugin**

**cURL : through unix command or git**

**Put() : update resource : update customer address, phone number using cid, update salary for employee using empId, update amount using accno.**

**Update query**

**Delete() : delete resource : delete employee records using id, order using id, customer using cid**

**Delete query**

**We have to pass the value using path param.**

**Employee**

**Customer**

**Account**

**Project**

**Customer : cid,cname,age,add{city,state}**

**CRUD Operation**

**Get() : get customer details using cid**

**If customer present display customer details else no customer present.**

**Get() : get all customer details.**

**Post() : create new customer details.**

**Put() : update age using cid, address (city, state)**

**Delete() :delete custom details using cid.**

**Create New folder CustomerExpressREST\_API**

**Create package.json file**

**npm init –y**

**npm install express body-parser**

**get method : through browser, hyperlink and submit button**

**post: we can call through submit button**

**put and delete : only through Rest client**

**browser plugin, Ajax, Angular, React JS or another programming language which support REST API.**

**Create the Angular project**

**ng new angular-customer-crud-operation**

**No routing:**

**ng g c search-customer-by-id**

**ng g c display-customer-details**

**ng g c store-customer-info**

**ng g c delete-customer-details**

**ng g c update-customer-info**

**ng g s customer**

**Angular running on port number 4200**

**Express JS running on port number 9090**

**Here two domain are communicating to each to each others.**

**CORS**

**Cross origin Resource Sharing : Backend technologies has restriction to call through front end technologies.**

**To enable cors features in Express js module we have to install external module as**

**npm install cors**

**07-08-2021**

**Rather than storing value in array we can store the records in file system using fs module.**

**File system is not a secure.**

**To apply query like insert, delete, update and retrieve consider as move complex call.**

**Data : raw fact**

**Information : processed data or meaningful data.**

**Database : storing the data in table format using row and columns.**

**DBMS : It is software which help to store the data in table format. Database Management system.**

**Excel sheet.**

**Database Model**

**Relational Model**

**If data base support 12 rules provided by Dr EF codd’s rules.**

**Database Name Company**

**MySQL Sun micro system (oracle)**

**Oracle oracle**

**Sql Server 2020 micro soft**

**Db2 etc IBM**

**They use common query language ie SQL. Which help to interact with database.**

**80 to 90% query is common but 10 to 20 % query get change when we move from one data base to another database.**

**RDBMS**

**Mongo DB database.**

**Mongo DB is known as No SQL Database.**

**No SQL Database**

**Key-value : redis**

**Graph database : Neo4j**

**Document Oriented : Mongo DB**

**Column family – Casandra or Hbase**

**Mongo DB store the data in document format.**

**Mongo DB database is a open source data which help to store the data in the json format using document.**

**Mongo DB database is known as schema less database.**

**RDBMS :**

**Schema :**

**Table -🡪Employee**

**Id Name Age PhoneNumber**

**100 Raj 21 9911**

**101 Raju 24 null**

**102 Ramesh 26 99000**

**103 Ram null 9911**

**Backend technologies may Spring boot(java), asp.net, Python or node JS.**

**REST API -🡪 backend technologies consumes the data in json format.**

**If we want to store the data in RDMBS data base we have to convert JSON data in table format.**

**Mongo db database is schema less and store the data in json format.**

**Banking Application**

**Customer :**

**cid,name,age,phonumber etc**

**RDBMS**

**Online shopping**

**No SQL Database**

**Product**

**Pid, Pname, price**

**Exp**

**Company**

**Kg/liter/**

**Install the mongo db database.**

**After installation**

**Open the home directory and open the command prompt in bin folder**

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

**mongod : This batch or application is use to run mongo db database.**

**Inside c drive create data folder and inside data folder create db folder**

**Then open another terminal in bin folder**

**Then to open the mongodb terminal run the command as**

**Mongo : mongo terminal to run DB queries.**

**Demo.txt**

**1 Raj 12000**

**2 Reeta 14000**

**Mongo DB queries**

**show databases : it display all pre-defined or user-defined databases.**

**show dbs : it display all pre-defined or user-defined databases.**

**My SQL**

**create database mydb : it use in mysql to create database but not in mongod**

**use mydb**

**Mongo DB**

**use databaseName : if database is present it switch to that database else it create and switch to that database.**

**In RDBMD database is known as group of tables.**

**In Mongo DB we store the data in collection.**

**Mongo db database is known as group of collections.**

**In RDBMS In Mongo DB**

**Database database**

**Table Collection**

**Row or records Document**

**Primary key (any column) \_id primary key**

**Mongo DB is case sensitive**

**Syntax to create the collection**

**db.createCollection(“Sample”)**

**To check all collection name present in database.**

**Show collections**

**Or**

**Show tables**

**Insert the data in collection.**

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

**key must be string and value may be number, string, Boolean, array type, complex object type.**

**To view document from collection**

**db.collection.find();**

**in mongo db every document store independently.**

**RDBMS by default doesn’t create primary key.**

**So in RDBMS we can store duplicate records if we didn’t create primary key.**

**In mongo db while inserting in document if we insert user-defined field. Mongo db internally create pre-defined field ie \_id as well as it store unique value using ObjectId pre-defined API.**

**Day 6**

**08-08-2021**

**Big Data :**

**Structure : Table**

**Mini / semi Structure - > XML, JSON etc**

**Un structure : video, audio, clips, images,**

**Graphic etc**

**Hbase**

**In Mongo DB we can insert the document in collection without creating collection also. If collection present in insert in existing collection else it will create new collection.**

**Create Emp collection and insert min 5 to 8 document with properties as \_id, name,age,city,state etc.**

**To retrieve the specific document from a collection using index position.**

**db.CollectionName.find()[indexPosition];**

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

**To retrieve the specific document fields or properties using index position.**

**db.CollectionName.find()[indexPosition].propertyName**

**Example**

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

**db.Emp.find()[2].age**

**retrieve one or more than one properties or fields from collection**

**db.collectionName.find({},{propertyName:1})**

**+ve or –ve number consider as true and 0 is false.**

**db.Emp.find({},{name:1}) : it display name as well as pre-defined property ie \_id.**

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

**retrieve more than one properties values using index position.**

**db.CollectionName.find({},{field:1,field:,\_id:0})[indexposition];**

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

**limit n number of records to display.**

**db.collectionName.find().limit(n);**

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

**skip n number of document from a collections.**

**db.collectionname.skip(n)**

**db.Emp.find.skip(3);**

**skip and limit**

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

**sort the document using specific property**

**db.CollectionName.find().sort({propetyName:1/-1})**

**+ve value asc and –ve desc order.**

**Retrieve the document from collection with conditions. Like a where clause in RDBMS.**

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

**to display the document with equal conditions.**

**db.Emp.find({\_id:100});**

**db.Emp.find({state:'Kar'});**

**db.Emp.find({name:'Ajay'});**

**conditional operator (numerical values).**

**> $gt**

**>= $gte**

**< $lt**

**<= $lte**

**= $eq**

**!= $ne**

**db.Emp.find({age:{$gt:25}})**

**db.Emp.find({age:{$lt:25}})**

**db.Emp.find({age:{$eq:25}})**

**$and / $or**

**$and means both condition must be satisfied**

**$or any one condition must be satisfied.**

**db.CollectionName.find({$and:[{property1:value},{property2:value2}]});**

**db.Emp.find({$and:[{\_id:100},{name:"Raj"}]});**

**db.Emp.find({$and:[{name:"Raj"},{age:21}]});**

**db.Emp.find({$and:[{state:"Kar"},{age:{$gt:20}}]});**

**$or : any condition must be satisfied.**

**db.Emp.find({$or:[{\_id:1000},{name:"Raj"}]});**

**db.Emp.find({$or:[{\_id:100},{name:"RajKumar"}]});**

**db.Emp.find({$or:[{\_id:1000},{name:"RajKumar"}]});**

**update query**

**db.collectionName.update({condition},{$set:{property:value}})**

**base upon one property update another one property**

**db.Emp.update({\_id:100},{$set:{age:22}});**

**base upon one property update more than another properties.**

**db.Emp.update({\_id:102},{$set:{name:"Ramesh Kumar",age:20}});**

**db.Emp.update({age:{$gt:25}},{$set:{city:"Bangalore",state:"Kar"}});**

**It update only one document.**

**By default if update condition satisfied for more than one document also, it update only one document.**

**So if you want to update more than one document then we have to use field as multi:true**

**db.Emp.update({age:{$gt:25}},{$set:{city:"Bangalore",state:"Kar"}},{multi:true});**

**Or**

**db.Emp.updateMany({age:{$gt:25}},{$set:{city:"Mysore",state:"Kar"}});**

**Remove**

**Remove document with one condition**

**db.Emp.remove({\_id:100});**

**Remove document with multiple condition satisfies**

**db.Emp.remove({$and:[{\_id:100},{name:"Ramesh Kumar"}]});**

**db.Emp.update({\_id:100},{name:"Raj Deep",age:32,salary:24000,city:"Bangalore",state:"Kar"});**

**This function check the \_id condition. If \_id present it replace old field or properties by new fields or properties else nothing happen by default.**

**db.Emp.update({\_id:106},{name:"Raj Deep",salary:24000,city:"Bangalore",state:"Kar"},{upsert:true});**

**This function check the \_id condition. If \_id present it replace old field or properties by new fields or properties else it insert new document.**

**Adding/Remove new fields for existing document.**

**db.Emp.update({\_id:103},{$set:{salary:12000}});**

**This function check with \_id property. If salary fields missing it add new field as salary with value as 12000. If salary field already exits then replace the value with existing fields values.**

**db.Emp.update({\_id:103},{$unset:{age:1}});**

**This query is use to remove the existing field from a document.**

**We have to work on more than one collections.**

**Collection Relationship**

**RDBMS : Relationship**

**Normalization**

**Normalization store data in different tables base upon the normalization.**

**Join : De-Normalization : which help to combine the records from more than one tables**

**We use primary key and foreign key connect two tables.**

**One to Many relationship 🡪 primary key and foreign key**

**Trainer --- Student.**

**Many to one relationship 🡪 primary key and foreign key**

**Employee --- Project**

**One to one relationship 🡪 Primary key to primary key**

**Person – Passport / PANCard**

**Many to Many relationship 🡪**

**Students ----Courses / Technologies**

**Employees ---- Projects**

**Primary key primary**

**Foreign key foreign key**

**Mongo Db maintain the relationship using two**

1. **Embedded collection**
2. **Linking collection**

**Embedded collection style (only one collection)**

**Employee**

**{id:100,name:”Raj”,salary:24000,age:21}**

**Address**

**{addId:123,city:”Bangalore”,state:”Kar”}**

**Employeedtails**

**Id Name Salary Age addId, city, state**

**100 Raj 24000 21 123 Bangalore Kar**

**Employee**

**PK FK**

**Id name salary addId**

**100 Raj 24000 123**

**Address**

**PK**

**addId City State**

**123 Bangalore Kar**

**Employee**

**{id:100,name:”Raj”,salary:24000,age:21}**

**Address**

**{addId:123,city:”Bangalore”,state:”Kar”}**

**{\_id:100,name:”Raj”,salary:24000,age:21,address: { city:”Bangalore”,state:”Kar”}**

**}**

**Adding complex array properties to existing documents.**

**db.Employee.update({\_id:1},{$set:{projects:[{pid:100,tech:"JEE"}]}});**

**updating complex properties values.**

**db.Employee.update({\_id:1},{$set:{“address.city”:”Mysore”});**

**Adding array complex property to existing documents.**

**db.Employee.update({\_id:1},{$push:{projects:{pid:101,tech:"Python"}}});**

**Collection relationship with linking style.**

**Trainer**

**PK**

**TId TName tech**

**1 Raj Java**

**2 Ravi Python**

**3 Ramesh C**

**Student**

**PK FK**

**Sid SName Age TSId**

**100 Seeta 21 1**

**101 Meeta 22 1**

**102 Veeta 23 2**

**103 Teeta 24 2**

**104 Keeta 25 null**

**Trainer collection**

**db.Trainer.insert({\_id:1,tname:"Raj",tech:"Java"});**

**db.Trainer.insert({\_id:2,tname:"Ravi",tech:"Python"});**

**Student1**

**db.Student1.insert({\_id:100,sname:"Seeta",age:21,tsid:db.Trainer.find()[0].\_id});**

**db.Student1.insert({\_id:101,sname:"Reeta",age:21,tsid:db.Trainer.find()[0].\_id});**

**Student2**

**db.Student2.insert({\_id:102,sname:"Meeta",age:22,tsid:db.Trainer.find()[0]});**

**db.Student2.insert({\_id:103,sname:"Leeta",age:23,tsid:db.Trainer.find()[1]});**

**db.Student2.insert({\_id:105,sname:"Beeta",age:25,tsid:[db.Trainer.find()[0],db.Trainer.find()[1]]});**

**Aggregate Function : aggregate function groups multiple document and then perform aggregate operation on it and it return a single result base upon the group. It is like a group by clause with aggregation operation like sum, max,min, count in RDMBS.**

**db.Employees.insertMany([**

**{\_id:1,name:"Ravi",age:21,salary:24000,deptId:10,city:"Bangalore"},**

**{\_id:2,name:"Ramesh",age:24,salary:14000,deptId:30,city:"Mumbai"},**

**{\_id:3,name:"Rajesh",age:25,salary:34000,deptId:20,city:"Delhi"},**

**{\_id:4,name:"Lokesh",age:30,salary:28000,deptId:10,city:"Bangalore"},**

**{\_id:5,name:"Ajay",age:26,salary:23000,deptId:10,city:"Bangalore"},**

**{\_id:6,name:"Vijay",age:36,salary:26000,deptId:20,city:"Mumbai"},**

**{\_id:7,name:"Balaji",age:32,salary:27000,deptId:20,city:"Mumbai"},**

**{\_id:8,name:"Dinesh",age:35,salary:30000,deptId:30,city:"Bangalore"},**

**{\_id:9,name:"Seeta",age:24,salary:28000,deptId:30,city:"Delhi"},**

**{\_id:10,name:"Meeta",age:22,salary:30000,deptId:10,city:"Bangalore"},**

**])**

**Group by Query**

**db.Employees.aggregate([{$group:{\_id:"$city"}}]);**

**db.Employees.aggregate([{$group:{\_id:"$age"}}]);**

**select sum(salary) from employee group by city,deptId**

**having clause**

**Banglaore**

**10**

**20**

**20**

**Delhi**

**20**

**30**

**30**

**Mumbai**

**Index : index use to improve speed or efficiency when we execute any query to search some document from a collections.**

**Show existing indexes**

**Non unique index**

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

**Unique index**

**db.Person.createIndex({phoneNumber:1},{unique:1});**

**Phase 3**

**15-08-2021**

**Node JS provide pre-defined external module**

**mongodb**

**mongoose**

**with the help of these modules we can connect to mongo db database through node js application.**

**Through Node JS we insert, delete, update and retrieve documents from collections.**

**Create the folder as mongodb module**

**Then install mongodb module using command as**

**npm install mongodb**

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

**and enter continuously and give the options yes at last.**

**After this command in current directory you can see package.json file. Package.json file is known as configuration file for node js application.**

**Difference**

**Installing any external module globally and locally.**

**npm install –g mongodb**

**npm install mongodb**

**if we install all modules locally package.json file hold all module details using dependencies and devDependencies key and value format.**

**After share the application to other with out node\_modules. There team has to execute**

**npm install**

**node search all dependencies details from package.json file and download in current machine.**

**npm i express or npm install express : locally npm i -g express or npm install –g express : globally npm i --save express or old version with help –-save flag providing module details in package.json. So new version –-save flag is optional.**

**npm i express --save-dev :**

**typescript or testing dependencies only required in develop mode not in production environment.**

**npm install jasmine --save-dev**

**npm uninstall express**

**mongoose : mongodb is a native module which help to connect the data. Mongoose is support data modelling concept. Using mongoose we can define schema for collection and base upon the schema we can store the document inside a collection.**

**Mongodb is like a jdbc**

**Mongoose is like a ORM.**

**Create new folder or directory mongoose module programs.**

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

**Please install**

**npm install mongoose**

**mongoose**

**we have to load the module.**

**Connect the database**

**Then we have to create the schema : which provide properties name and with data type.**

**Base upon the schema we have to create the model class.**

**Using model we can do the operation on collection may be insert, delete, update and retrieve.**

**MVC : Model View Controller**

**Express MVC :**

**http module : view html page**

**express js also provided lot pre-defined view plugings.**

**Jade**

**If we use express js any html plugin our view become tightly coupled. So After Rest API. View is**

**Angular, React JS or any other technologies.**

**Express**

**View 🡪 Angular -🡪 app.js-🡪router.js 🡪controller.js🡪model.js --🡪 Mongo DB Database**

**Front end using Angular**

**Html(template)-🡪component-🡪 service -🡪using HttpClient send to backend technologies 🡪**

**Backend using node js**

**App.js : load all the modules required to express REST API.**

**Express, body-parser, cors, mongoose etc.**

**Create the reference of express js**

**Connect the database.**

**Add middleware cors origin features.**

**Add middleware body-parser to enable body data from req part.**

**//** [**http://localhost:9090/api/product**](http://localhost:9090/api/product)

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

**//** [**http://localhost:9090/api/login**](http://localhost:9090/api/login)

**Add router file path which help to check main path url available in rest API. So base upon path it will re-direct to specific router file.**

**Then run the application on specific port number.**

**router.js**

**produt.router.js**

**get**

[**http://localhost:9090/api/product/getAllProduct**](http://localhost:9090/api/product/getAllProduct)

**post**

[**http://localhost:9090/api/product/storeProduct**](http://localhost:9090/api/product/storeProduct)

**customer.router.js**

**login.router.js**

**This file receive the request from app.js and check sub path and specific method like get, post, put and delete. Base upon method and sub path it will pass the request to controller.**

**Controller.js**

**Product.controller.js**

**We can write set of methods do to operation one collection base upon the sub path provided by router file.**

**Customer.model.js**

**Login.model.js**

**Controller take the help of model file and do the operation on collection base upon the request receive from router.**

**Model.js**

**product.model.js**

**customer.model.js**

**login.model.js**

**Model file load the mongoose module create the schema. Using Schema we have to create model with collection name.**

**MEAN : Mongo DB Express angular and Node**

**Employees**

**id, name, salary**

**Create folder MVC**

**Inside MVC create two folder**

**Backend**

**Frontend**

**Backend**

**Create package.json file using command as**

**npm init –y**

**Or**

**npm init**

**npm install express cors mongoose**

**error code**

**100 : information**

**200 : success**

**300 : re-direct**

**400 : client side error**

**500 : server side error**

**22-08-2021**

**ng new employee-crud-operation**

**ng g c employee : it is use to create component**

**ng g s employee : it is use to create the service**

**ng g class employee : it is use to create model class**

**RxJS : Reactive JS**

**Provide set of operator which help to do some operation subscribe data.**

**If you want to use any operator we have to use pipe() function and we can use more than one operator.**

**Phase 3**

**28-08-2021**

**Authentication and Authorization**

**Socket IO**

**networking : net module**

**web Socket**

**socket.io**

**Networking : sending the data from one machine to another machine within a same network using TCP protocol. TCP (Transmission Control Protocol).**

**Http or Https**

**One machine another machine**

**Using API send**

**Data received or failure**

**Non Node JS application data can get block.**

**Node JS help use to send and receive non blocking IO operation through network environment.**

**Node js provide pre-defined module ie net which help to send the data from one node(machine) to another node(machine).**

**Client Server**

**TCP machine machine**

**Program program**

**Within one net work**

**Environment.**

**Http machine machine**

**Browser server**

**Access web application using http protocol get response in html format.**

**Server.js**

let net = require("net");

let server = net.createServer();

**servet.listen(9000,()=>console.log(“Server running on port number 9000”))**

**if you run this application server running on port number 9000.**

**To connect this application using another machine or node. Make sure both machine under one network environment.**

let net = require("net");

let client = net.createConnection(9000,"127.0.0.1",()=> {

    console.log("Client send the request to server");

    // Sending the message from client to Server

    client.write("Hello Server! I am Client Application");

})

**Client application has to load the module and create the connection with port number and ip address of server application machine. If server running on same machine then you can give localhost or 127.0.0.1. if server running on different port number then you have to provide ip address of that machine.**

**Using socket we are connect two machine or device or node to share the data from one machine to another machine.**

**To do this one every programming language provide api**

**Like node js provided net module. Which help to do socket programming.**

**Web Socket : Socket programming on web application.**

**In socket programming both application are standalone or desktop application develop in any language.**

**Web Socket provide Pre defined API which help do two-way interactive communication between browser and web application.**

**Web socket is a API or technologies. Socket.io is a library base on web socket which done abstraction.**

**Hiding the internal implementation of logic.**

**JavaScript jQuery**

**--------------------🡪**

**Client Server**

**🡨------------------**

**Push notification or advertisement.**

**To achieve web socket features using node js we have to install two module**

**npm install expess : web application**

**npm install express-ws : web application with web socket.**

**Create websocket programming folder**

**Using package.json file using**

**npm init**

**Install two modules**

**npm install express**

**npm install express-ws**

**..To achieve socket programming on web application JavaScript provide pre-defined it WebSocket.**

**http protocol one way communication**

**ws protocol two way duplex communication.**

**let ws = new WebSocket(“ws://localhost:9090”);**

**Socket.io : Socket.io is a library. It enables real time bi directional event base communication.**

**Socket.io internally use WebSocket API to do bi- directional communication.**

**In socket.io we are going to use three modules**

**http**

**express**

**socket.io**

**create folder with socket io program**

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

**npm init**

**then install two modules**

**npm install express socket.io**

**Rather than open index page**

**Open first page as login page and take the username**

**Using queryparam and pathparam**

**Authentication and Authorization:**

**Authentication is use to connect the application with login authentication details.**

**Authorization mainly use to access services in that application. Like add, remove, update display.**

**In simple word, authentication is the process of verifying who is?**

**Authorization is the process of verifying what they access in that application.**

**Admin user : can do all operation in that application**

**Normal user : can only view or order that product.**

**Now way day we are using JWT ( Json Web Token).**

**Http protocol is stateless protocol. Using http protocol we can’t do session tracking concept.**

**1st req : static req**

**2nd req login + sessionId**

**3rd req + sessionId**

**4th req + sessionId**

**Client Server**

**1st login page static res**

**2nd res success page home open**

**+ cookies file**

**Cookies : it is small text file create by server when client send the request to server.**

**This file contains unique id ie sessionId.**

**File created by server but store in client machine.**

**Drawback cookies. Cookies not secure.**

**My be client doesn’t allow to store cookie file in their machine.**

**JWT : JSON Web token**

**1st Req : login page open**

**2nd req with username**

**3rd req + signed token**

**4th req + signed token**

**And password**

**Client Server**

**1st res : login page with response**

**2nd res + success + signed JWT token**

**3rd res + singed token**

**Jwt token present in http request header part.**

**JWT Structure :**

**Product related task**

**App1.js : 9091 router, controller, model**

**Customer related task**

**App2.js : 9092 router, controller, model**

**Login related task**

**App3.js : 9093 router, controller, model**

**Service : Java**

**Service : .net**

**Service : python**

**Create folder authentication and authorization**

**Create package.json file**

**Using npm init**

**Please install all these modules**

**npm install express**

**npm install cors**

**npm install mongoose**

**npm install jsonwebtoken : this is use to create the token**

**npm install bcryptjs : this use to store your password in bcrypt format.**

**Now create three folder :**

**controller, model and router.**

**1st req**

**2nd req + login Details**

**3rd req in dashboard to do some operation + token pass(token in header part)**

**Client Server**

**1st Res (static web page)**

**2nd response with success message server provide you token.**

**3rd response + token**

**Product : \_id pname price**

**Admin : can add and view product**

**User : can view only product**

**After created whole application.**

**First do the register**

**Method post**

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

Set header : content-type : application/json

Data

{

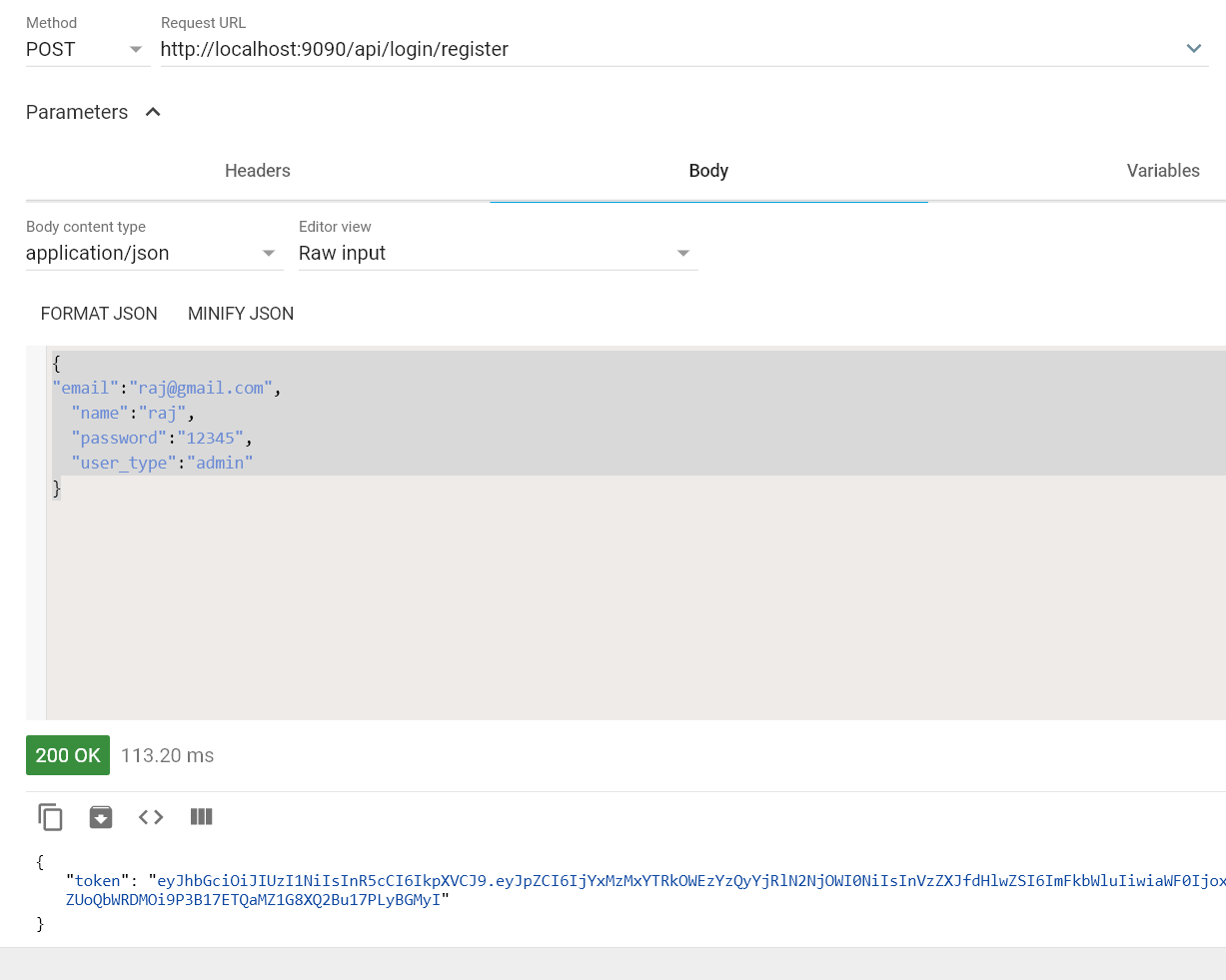
"email":"raj@gmail.com",

"name":"raj",

"password":"12345",

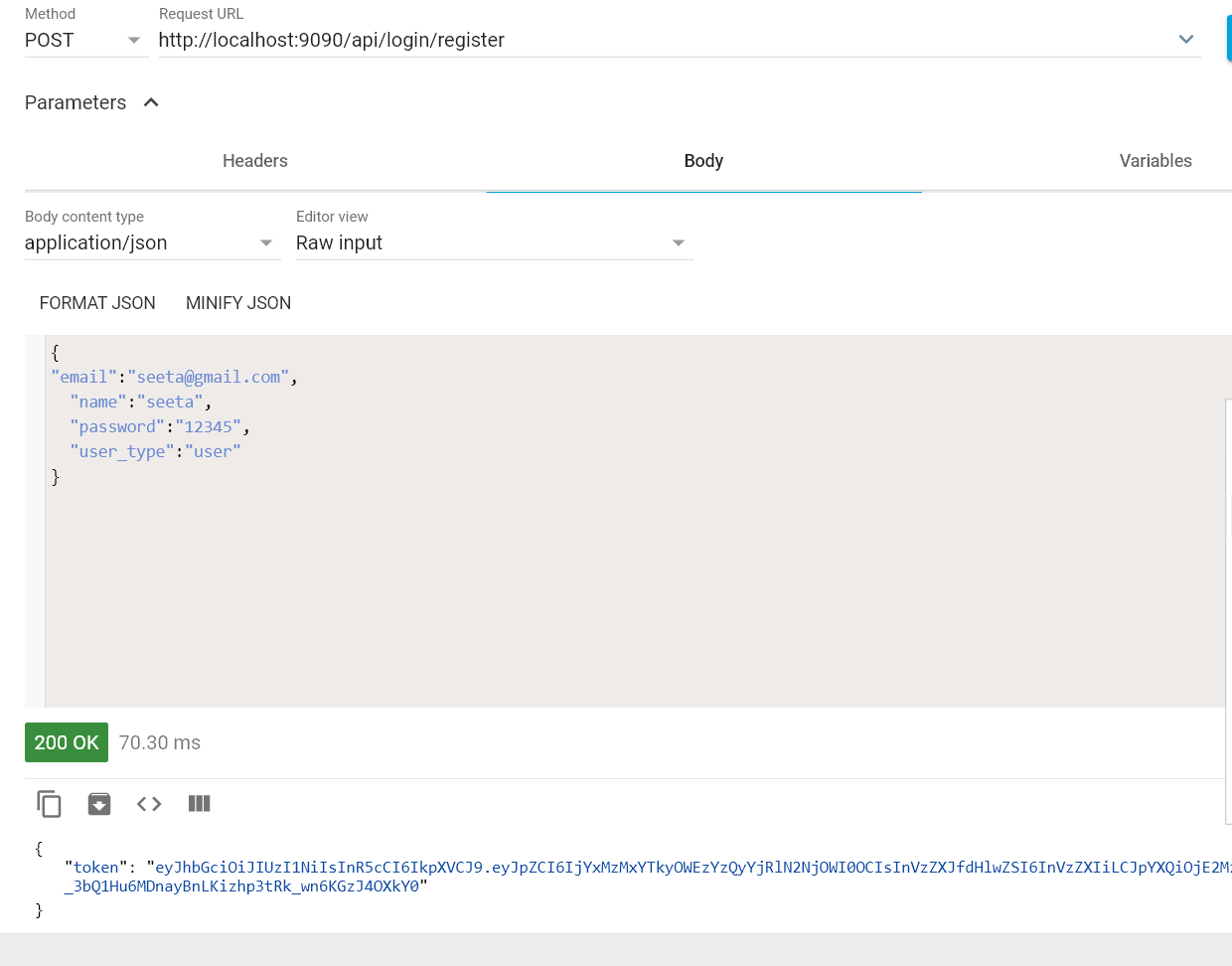
"user\_type":"admin"

}

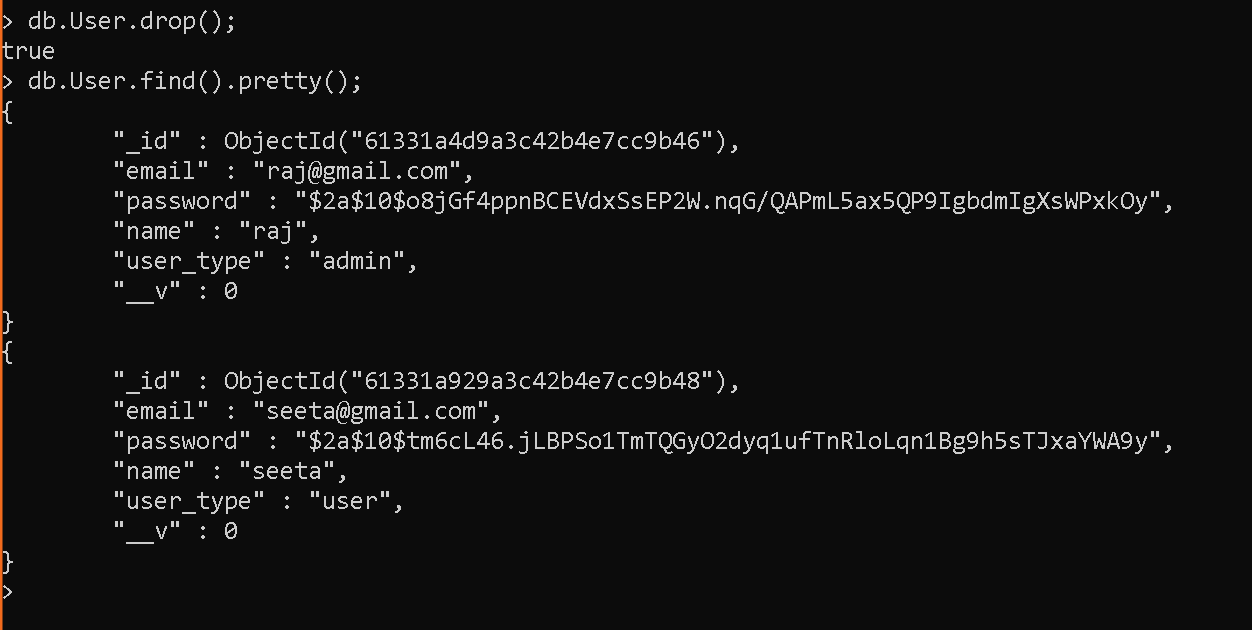


Account create for admin user

Now we will create user account



Please verify user collection in database.



Now check the admin account

Method post

<http://localhost:9090/ap/login/login>

header : content-type : application/json

data

{

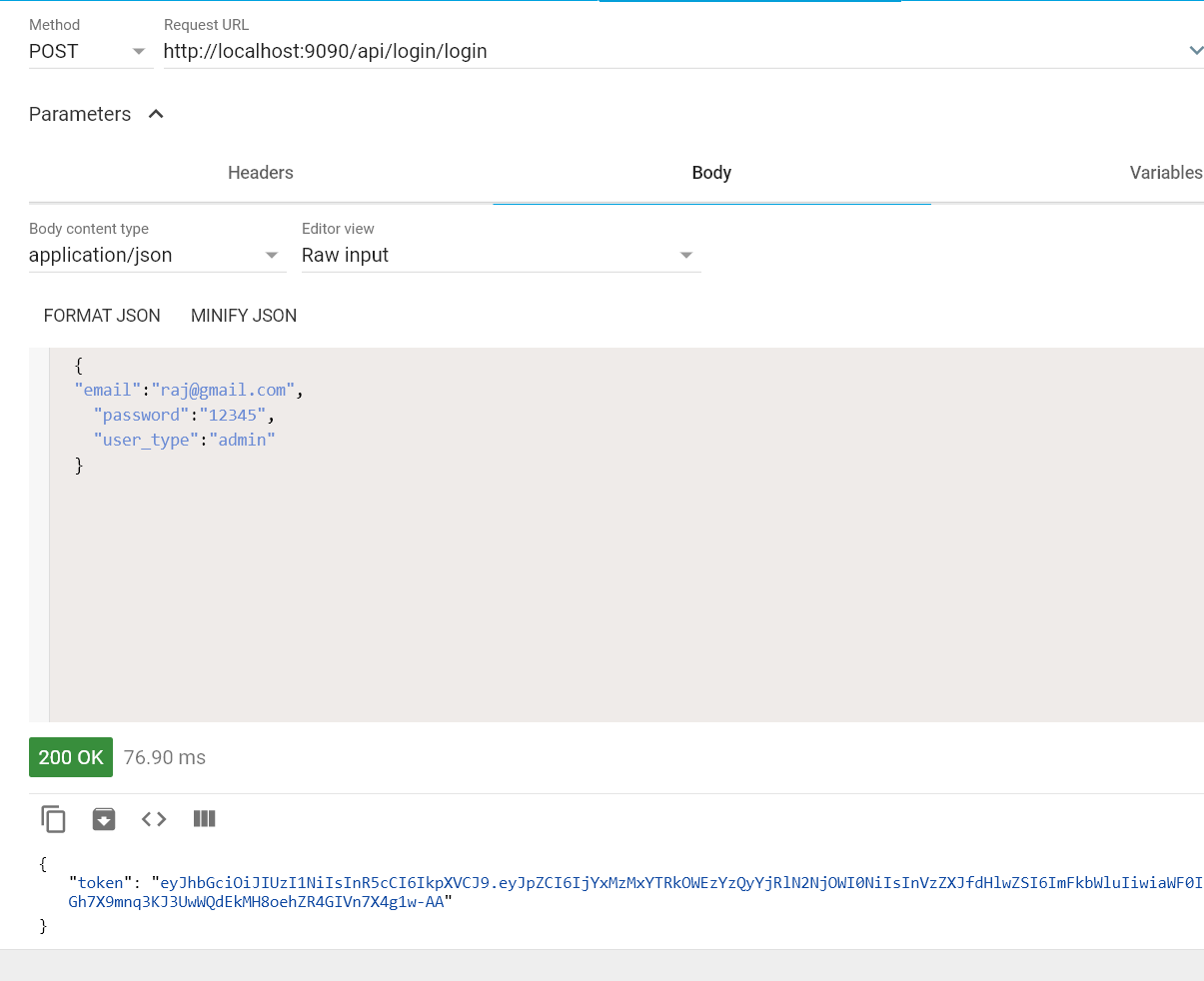
"email":"raj@gmail.com",

"password":"12345",

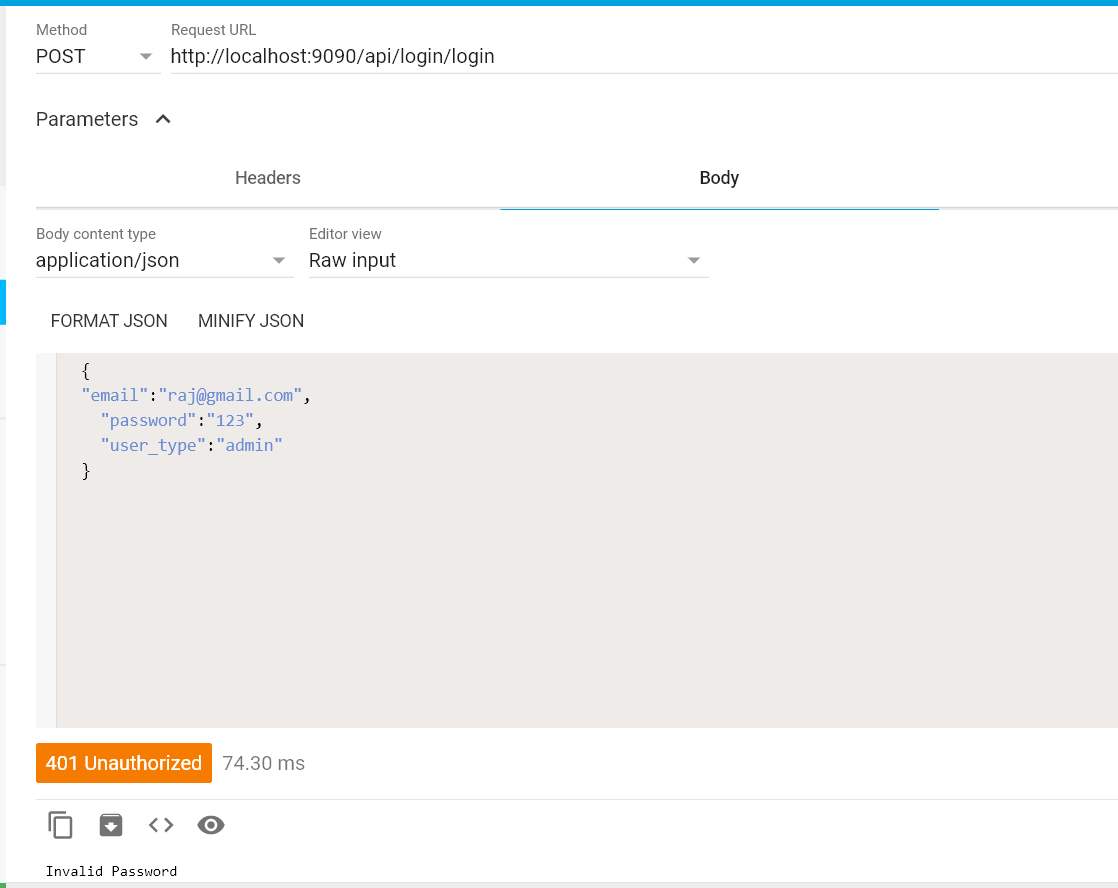
"user\_type":"admin"

}

**If email id and password is correct joken generate**

****

**Try with wrong user name or password.**

****

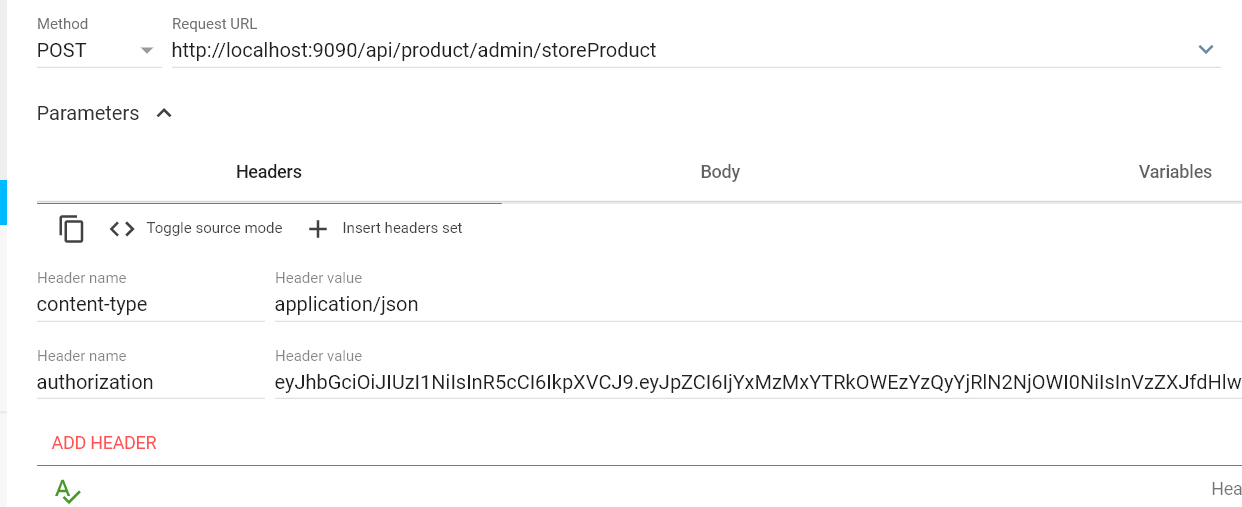
**After successfully login using admin user copy the token and try to access two REST API**

**Ie store product and retrieve project**

**While calling please add token in header part using authorization property.**

**Add product url is**

**http://localhsot:9090/ap/product/admin/storeProduct**

****

**In body part add the data in the fomr of json**

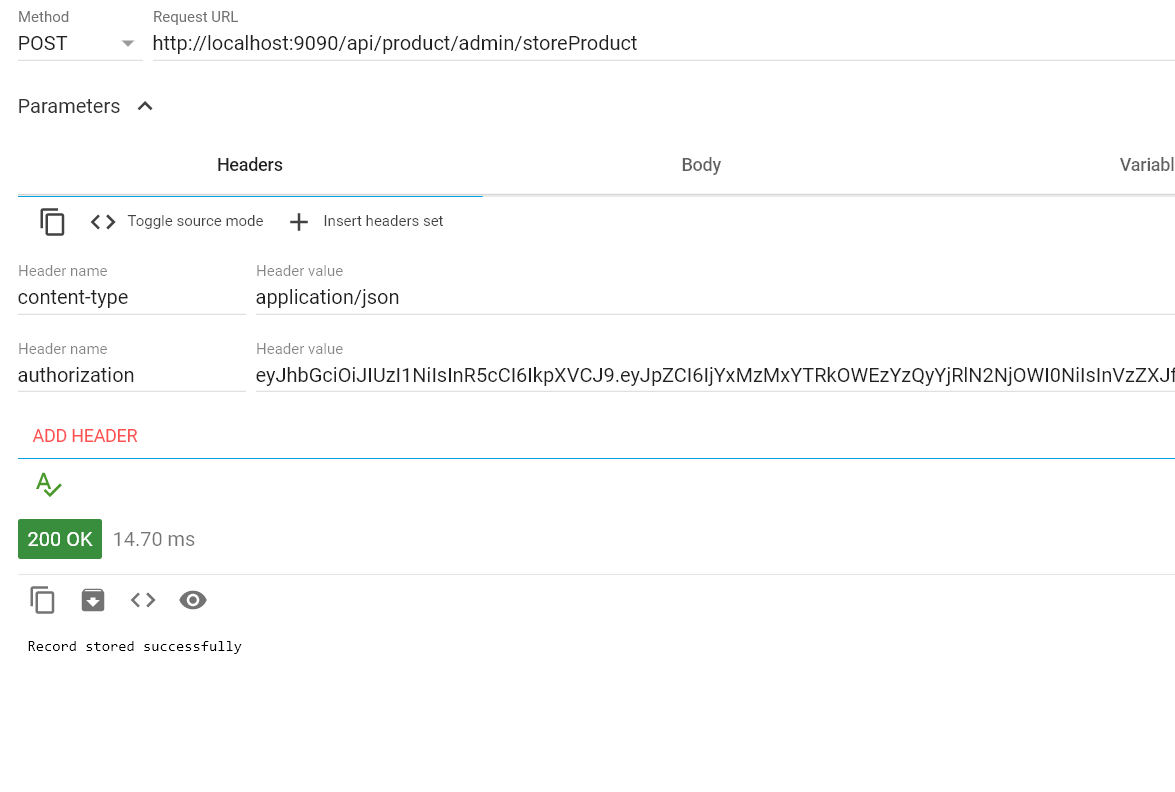
**{**

**“\_id”:1,**

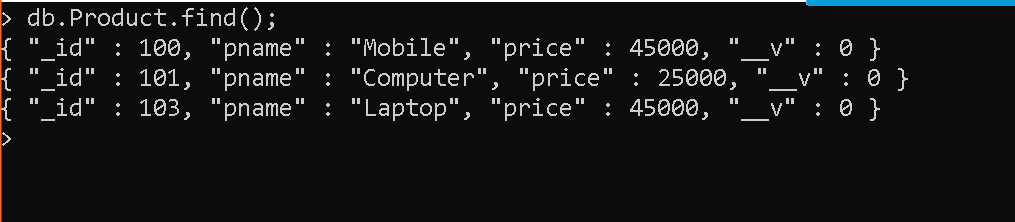
**“pname”:”TV”,**

**“price”:120000**

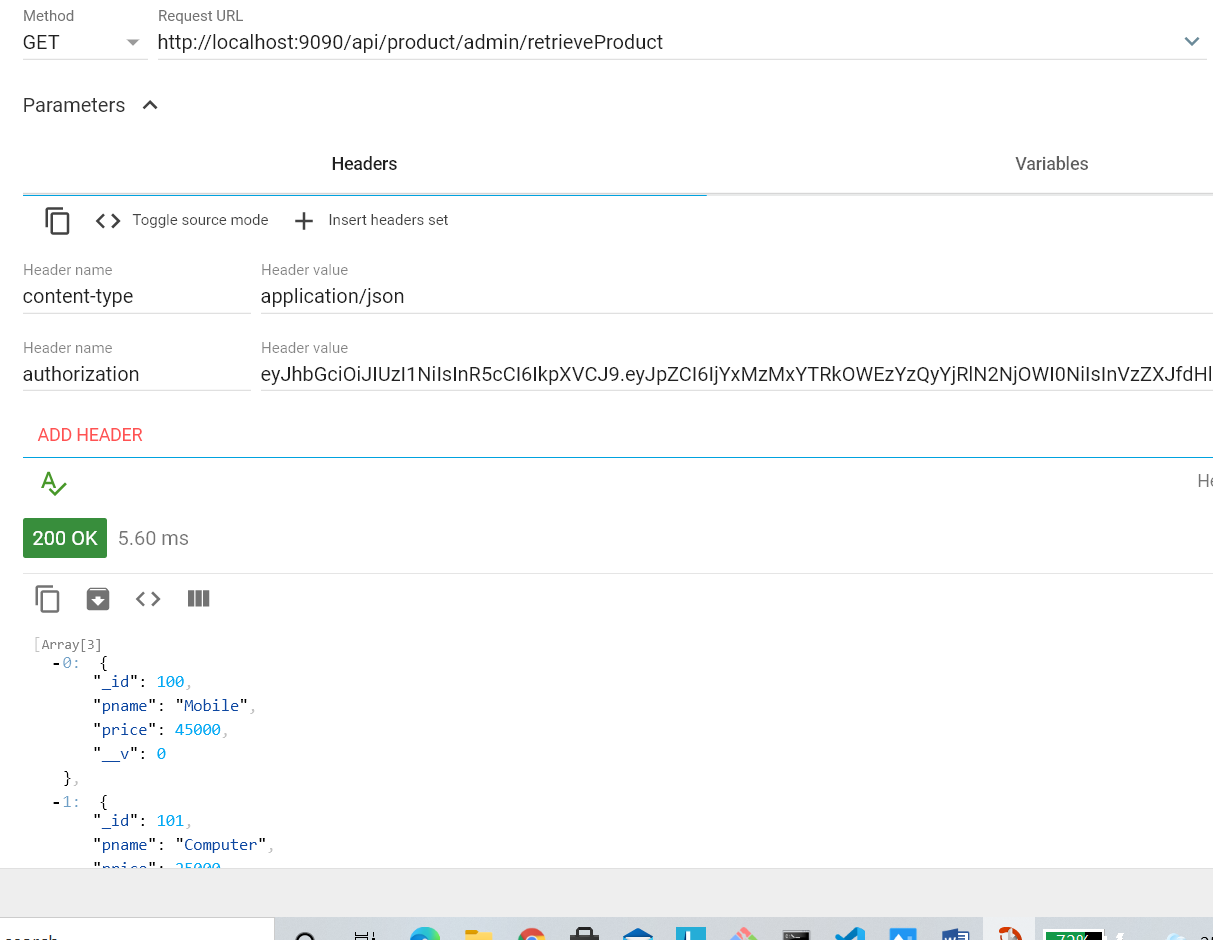
**}**

****

**After store more records please verify**

****

**Then try to retrieve records using api**

****