MERN STACK => MongoDb Express React Node

Express js => web framework for node, bulit on top of http module,

Express.static 🡪

* Allows to use middle wares to respond http requests
* Defines routing the table based on http methods
* Render html pages

Need to be installed 🡪 npm install express

For Post -

* Npm install body-parser
* Var bodyparser = require(‘bosy-parser’);
* Var urlencodedparser = bodyparser.urlencoded({})

Task 🡪

Use express to work with customer details(code,name,phone,address).

Create an api to get and post methods

Create appropriate forms for getting details from the user.

Step – 1:

Npm install express

Npm install body-parser

MongoDB –

RDBMS 🡪 fixed schema/table structure,

Oracle, MySQL, SQL Server …

=> Databases -> tables -> rows/columns

=> Primary key

NoSQL 🡪

MongoDB is a NOSQL, is document oriented

=> databases -> collection -> document/field

- platform independent

- open source

- high performance, availability, scalabilty

- data stored, json/bson format

- schema less

- Primary key (autogenerated field \_id,object id, is unique)

C:\> data\db

C:\> mongod.exe –dbpath “c:\data”

In server ternimal

In mongo(client) terminal:

Show dbs

Show collections

Use brillio

Use firstcollection

Switched to be firstcollection

Db

Db.employee.insert({“name”:“Naveen”, “age”:28})

Db.employee.insert({“name”:“Naveen”, “age”:28, “city”:”Nellore”})

Example:

Db.employee.insertMany(

[

{

First\_Name: “Naveen”,

Last\_Name: “Kumar”,

E\_id: “Naveen@gmail.com”,

Phone: 0987654321,

City: “Nellore”

},

{

First\_Name: “Praveen”,

Last\_Name: “Kumar”,

E\_id: “Praveen@gmail.com”,

Phone: 0987654321,

City: “Hyderabad”

},

{

First\_Name: “Divyesh”,

Last\_Name: “Kumar”,

E\_id: “Divyesh@gmail.com”,

Phone: 0987654321,

City: “Mumbai”

}

]

)

Beautify and pretty the db --

Db.employee.find().pretty();

Finding the details in db

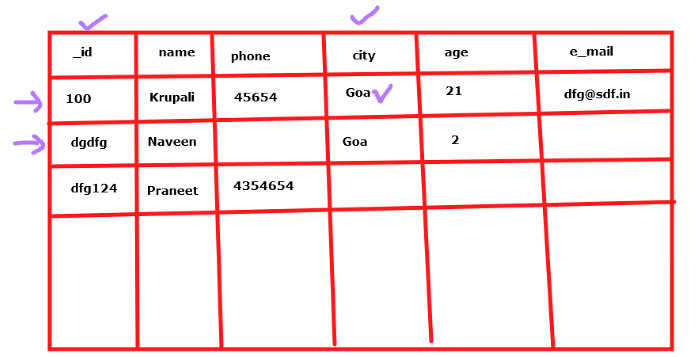
Db.employee.fin({city:”Nellore”});

Db.employee.find({city:”Nellore”}).pretty();

Db.employee.find({city:”Nellore”},{name:”Naveen”}).pretty();

Db.employee.find({city:”Nellore”},{name:”naveeen”, e\_id:”Naveen@gmail.com”}); //select name,e\_id from employee where city=”Nellore”

db.employee.find({City:"Mumbai"},{name:1, E\_id:1});



Db.employee.find({city:{$ne:”Munbai”}});

Db.employee.find({age:{$gt:10}});

Db.employee.find({age:12})

Db.employee.find({$or:[{age:12},{city: “hyderabad”}]})

Db.employee.find({age:{$gt:5, $lte:25}}) //age>5 and age<=25

Db.employee.find({age: 20}).count();

Db.employee.distinct(“city”);

Db.employee.update({\_id:101},{$set:{age:22}})

Db.employee.updateMany({city:”Mumbai”},{$set:{city:”Delhi”}})

Db.employee.remove({\_id:102},1); //2nd parameter is suggests the 1st matching document

Db.employee.remove({});

Db.employee.deleteMany({age:22})

Db.employee.deleteMany({});

**Working with Mongoose** :

Mern => React Ui -> Node Api -> MongoDB

Mongoose is a js library

Is a object data modeling(ODM) Library => creating some objects to map with documents of MongoDB

Steps:

1. Npm install mongoose –save
2. Import
3. Make a connection => mongoose.connect(mongodburl)
4. Define a schema => mongoose.schema({id:string, name:string, age:number})
5. Compile schema to model => var Employee =mongoose.model();
6. Create an instance of a document => var empObj =new employee({id:”1”, name:”Naveen”, age:28});
7. Save the model to the database => empObj.save();

**Task – 20-08-2021**

1. Create an api in expres with get and post method

* We need to install cors and use in express js file

1. Create a dummy product componnetn in sports app, use fetch/axios to call the get and post methods

**Micro Service --** Small Services

It is a architectural style. We implement it by creating many api’s

Earlier:

Monolithic Architecture, single Api (service) it’s a, containing code to execute all the features

Now:

It’s an architecture style -> collection of many api that are

-- highly maintainable and testable

-- Loosely coupled

-- Independently deployable

-- owned by small teams

-- organized around the business needs

Ex: Uber

* Loin Details -- user details
* Maps - Locations, tracking
* Boking of acab
* Types of cabs
* Payment options, billing
* Chat
* Driver Details
* Support

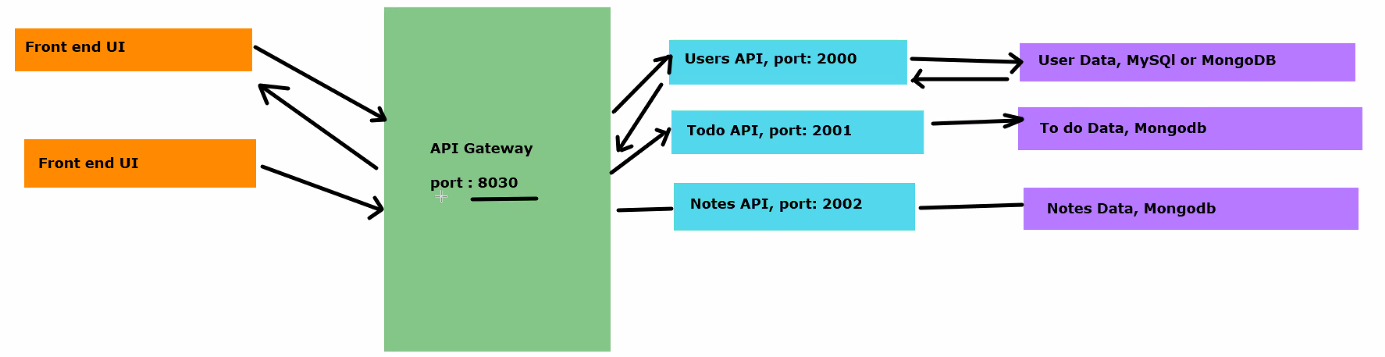
To Do App:

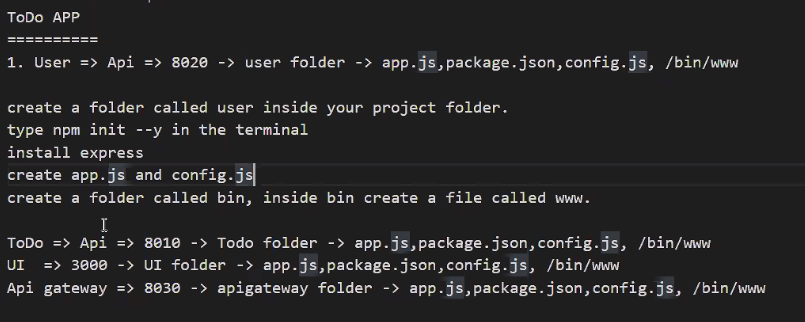
User Details 🡪 Crud 0n User 🡪 Api 🡪 2000

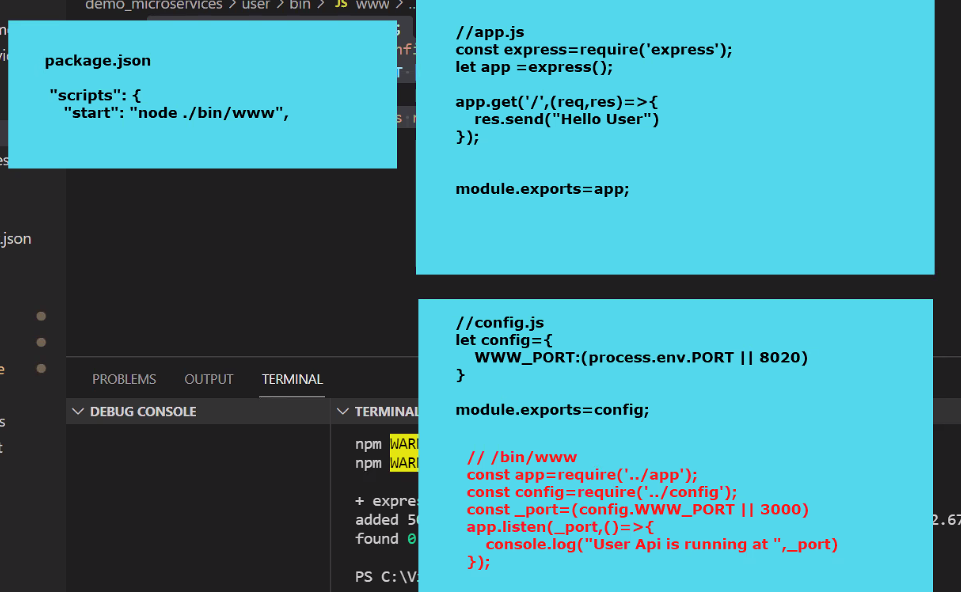
Todo Details 🡪 Crud on Todo 🡪 Api 🡪 2001

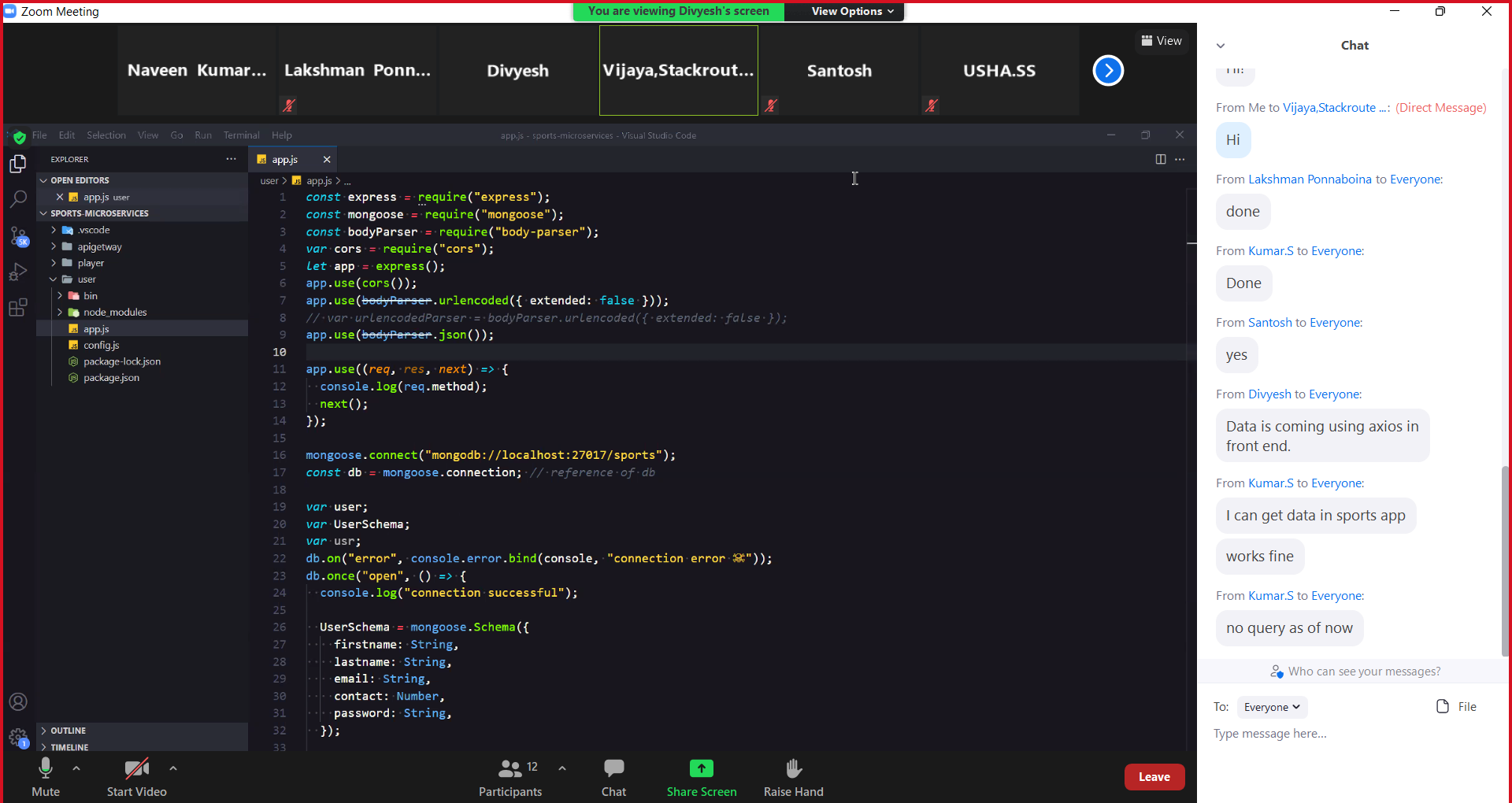
Notes Details 🡪 Crud on Notes 🡪 Api 🡪 2002

Api GateWay 🡺 8030



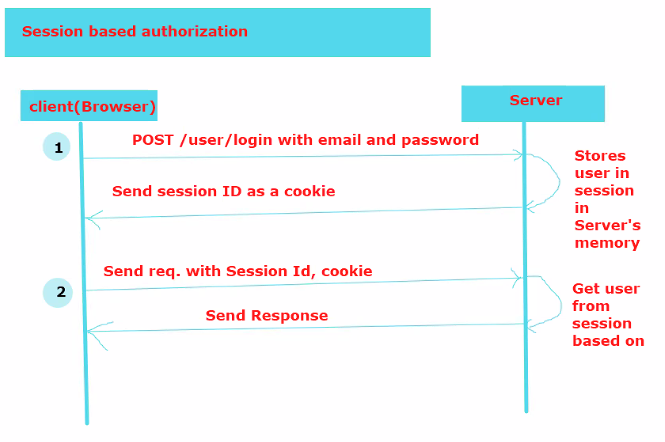


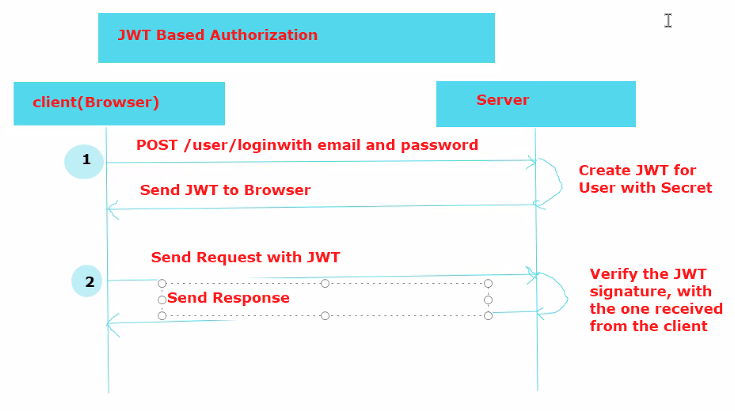




**Working with Json web Token (JWT) –**

JWT is used for authorization



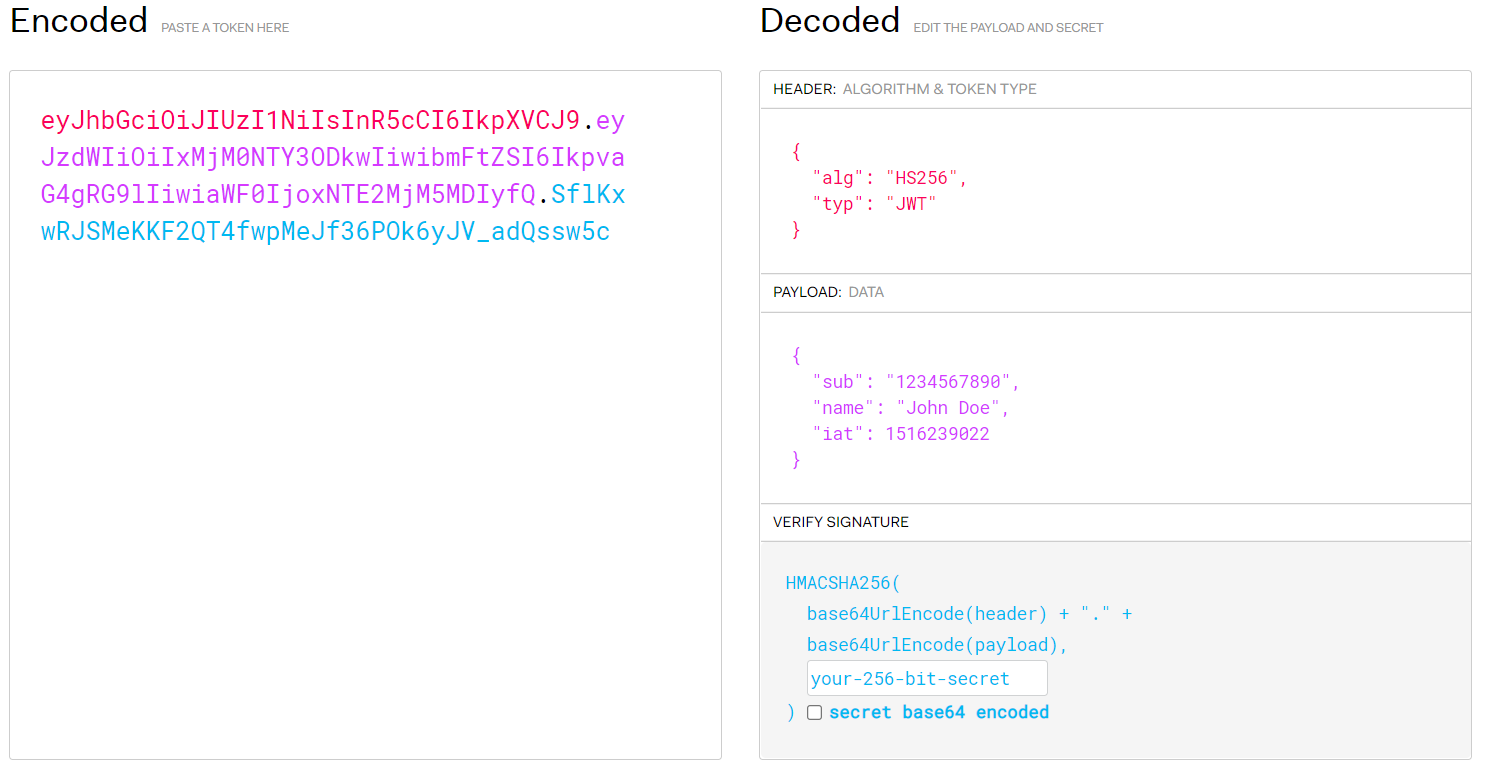


**JWT is used for authorization –**

JWT contains: header, payload, signature

Npm install jsonwebtoken dotenv

Npm i –save-dev nodemon



What are the steps happen in jwt

Why you compare the jwt better than session

**Docker—**

Working with docker => containerization

**SDLC**

Analysis

Planning

Design

Development

Deployment =>Docker operates here

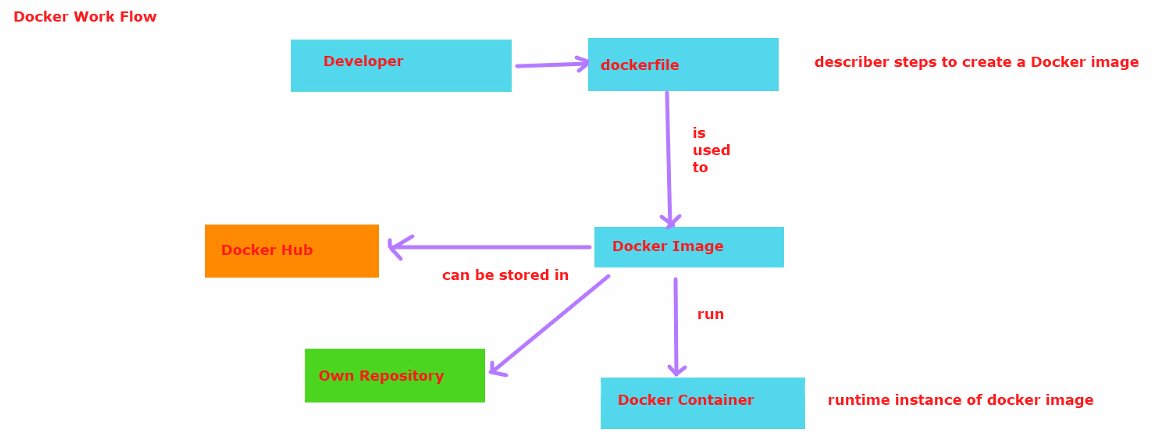
Testing / release

It makes the progress of deployment easy, efficient, resolves a lot of deployment related issues.

What is docker?

It is software container platform

Containers allow a developer to package up an application with all the needs like libraries, dependencies etc... ship it all of them as one package.

****