**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**MongoDB Advanced Operations**

**Capstone Project Backend Implementation**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

=================================================

MongoDB Advanced Operations

=================================================

Find particular record/s ?

>db.products.find(

{$where:function()

{

return(

this.p\_name === "P\_five"

)

}

}

)

Sorting mongodatabase

>db.products.find().sort({p\_name : 1})

Where p\_name is sort key

1 -> ascending order

-1 -> descending order

pretty() method

>db.products.find().pretty()

>db.products.find().sort({p\_name:1}).pretty()

Limiting Records

>db.products.find().limit(2)

where we can access only two records

MongoDB task

Create a mongodb database namely college

create collection Student

insert atleast 20 documents

Student Name

Branch

percentage

Gender

1. Find topper of CS branch

2. Find College topper

3. Find Topper from girls

4. Find Topper from boys

5. Sort according to percentage

6. Find Top 3 students

7. Find slow learners (less percentage)

{"name": ,"branch": , "percentage": , "gender": }

>use college

>db.createCollection("students")

>db.students.insertMany([{},{},{}])

1. db.student.find({$where:function(){return (this.branch == 'CS')}}).sort({percentage : -1}).limit(1)
2. db.student.find().sort({percentage : -1}).limit(1)
3. db.student.find({$where:function(){return (this.gender == 'female')}}).sort({percentage : -1}).limit(1)
4. { "\_id" : ObjectId("6367ccfc2f307318ca78c80c"), "name" : "Suresh Kale", "branch" : "CS", "percentage" : 80, "gender" : "male" }
5. db.student.find().sort({percentage : -1})
6. db.student.find().sort({percentage : -1}).limit(3)
7. db.student.find().sort({percentage : 1}).limit(3)

**Capstone Project POC**

- Design e-commerse website

- options are (two or more)

- Medicins

- Clothing

- Workout

- Mobiles

- Mobile accessories

- Computer equipments

- Computer accessories

- Books

- Groceries

- Car accessories

- Jewelleries

- watches

- Sun glasses

Backend

Functionality

- on sign in create a new user in users

- on login compare username and password with database and proceed accordingly

- on add to cart insert a record in cart collection

- on buy now update total quantity in products collection

API calls

- Create user -> insertuser

- Login -> login

- Show all products -> fetch

- Add to cart -> insertproduct / updateproduct

- Reduce from cart -> updateproduct / deleteproduct

- Buy now -> updateproducts

collections

- products

[

{

product\_1 details

},

{

product\_2 details

},

...

]

- users (uname, upwd)

[

{

user\_1 details

},

{

user\_2 details

},

...

]

- cart

[

{

id:

user:

product\_details:

},

...

]

Front end

- Website should have rich user interface

- There should be card layout for each product

- on mouse hover on product image that product should be enlarged

- There should be 'learn more', 'add to cart' and 'buy now' options

- There should be login page for sigining in

- There should be cart page where we will get number of added items

- In buy now page show complete cost of purchase

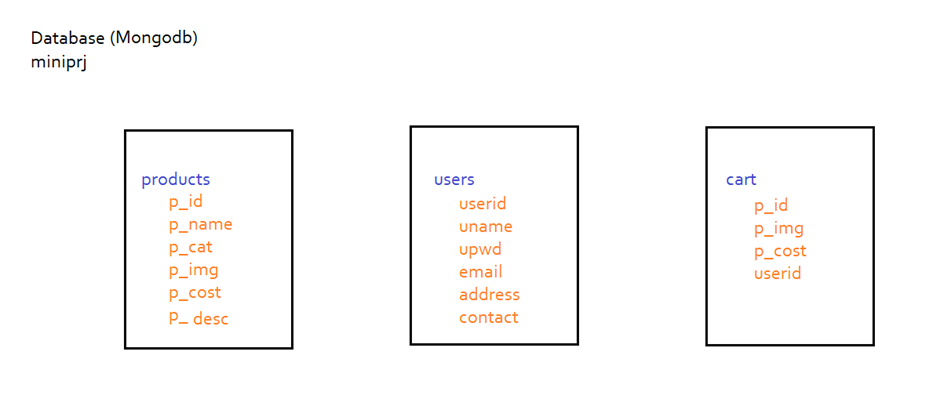
Enhancements (Optional)

- Try user interface without using Bootstrap CDN

- Use diffrent GST rates for various products

**Backend Functionality**

Create mongodb database ‘miniprj’



Create three collections

products

p\_id

p\_name

p\_cat

p\_img

p\_cost

p\_desc

users

userid

uname

upwd

email

address

contact

cart

*Use mongodbcrud code and update the same, dont change url.js and server.js*

\*\*\*fetch.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.get("/", (req, res) => {

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db('miniprj')

db.collection('products').find().toArray((err, array) => {

if (err)

console.log('Error :- ' + err)

else {

console.log('Data sent')

res.json(array)

conn.close()

}

})

}

})

})

//User login Authentication

router.post("/auth", (req, res) => {

let uname = req.body.uname

let upwd = req.body.upwd

let obj = { uname, upwd }

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db("miniprj")

db.collection('users').find(obj).toArray((err, array) => {

if (err)

console.log(err)

else {

if (array.length > 0)

res.json({ 'auth': 'success', 'user': uname })

else

res.json({ 'auth': 'failed' })

console.log('Auth response sent')

conn.close()

}

})

}

})

})

//Fetch cart data

router.post("/fetchCart", (req, res) => {

let uname = req.body.uname

let obj = { uname }

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db("miniprj")

db.collection('cart').find(obj).toArray((err, array) => {

if (err)

console.log(err)

else {

res.json(array)

console.log(`Cart response for ${obj.uname} sent`)

conn.close()

}

})

}

})

})

//export router

module.exports = router

\*\*\*insert.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.post("/", (req, res) => {

let obj = req.body

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection :- ', err)

else {

let db = conn.db("miniprj")

db.collection('products').insertOne(obj, (err) => {

if (err)

res.json({ 'insert': 'Error ' + err })

else {

console.log("Data inserted")

res.json({ 'insert': 'success' })

conn.close()

}

})

}

})

})

//Insert User

router.post("/createUser", (req, res) => {

let obj = {

"userid": req.body.userid,

"uname": req.body.uname,

"upwd": req.body.upwd,

"email": req.body.email,

"address": req.body.address,

"contact": req.body.contact

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection :- ', err)

else {

let db = conn.db("miniprj")

db.collection('users').insertOne(obj, (err) => {

if (err)

res.json({ 'userInsert': 'Error ' + err })

else {

console.log("User inserted")

res.json({ 'userInsert': 'success' })

conn.close()

}

})

}

})

})

//insert product into cart

router.post("/cartInsert",(req,res)=>{

let obj = {

"p\_id" : req.body.p\_id,

"p\_cost" : req.body.p\_cost,

qty : 1,

"p\_img":req.body.p\_img,

"uname" : req.body.uname

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection :- ', err)

else {

let db = conn.db("miniprj")

db.collection('cart').insertOne(obj, (err) => {

if (err)

res.json({ 'cartInsert': 'Error ' + err })

else {

console.log("Prouct in Cart inserted")

res.json({ 'cartInsert': 'success' })

conn.close()

}

})

}

})

})

//export router

module.exports = router

\*\*\*update.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.post('/', (req, res) => {

let p\_id = req.body.p\_id

let obj = {

"p\_name": req.body.p\_name,

"p\_cost": req.body.p\_cost

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db("miniprj")

db.collection("products").updateOne({ p\_id }, { $set: obj }, (err, result) => {

if (err)

res.json({ 'update': 'Error ' + err })

else {

if (result.matchedCount != 0) {

console.log("Data updated ")

res.json({ 'update': 'success' })

}

else {

console.log("Data Not updated ")

res.json({ 'update': 'Record Not found' })

}

conn.close()

}

})

}

})

})

//Update product in cart

router.post("/updateCart", (req, res) => {

let p\_id = req.body.p\_id

let uname = req.body.uname

let obj = { "qty": req.body.qty }

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db('miniprj')

db.collection('cart').updateOne({ p\_id, uname }, { $set: obj },

(err, result) => {

if (err)

res.json({ 'cartUpdate': 'Error ' + err })

else {

if (result.matchedCount != 0) {

console.log(`Cart data for ${uname} updated`)

res.json({ 'cartUpdate': 'success' })

}

else {

console.log(`Record not updated`)

res.json({ 'cartUpdate': 'Record Not found' })

}

conn.close()

}

})

}

})

})

//Update user

//to be done by participants

//export router

module.exports = router

\*\*\*delete.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.post("/", (req, res) => {

let obj = {

"p\_id": req.body.p\_id

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db('nodedb')

db.collection('products').deleteOne(obj, (err, result) => {

if (err)

res.json({ 'delete': 'Error ' + err })

else {

if (result.deletedCount != 0) {

console.log('Data deleted')

res.json({ 'delete': 'success' })

}

else {

console.log('Data Not deleted')

res.json({ 'delete': 'Record Not found' })

}

conn.close()

}

})

}

})

})

//Delete product from cart

router.post("/deleteCart", (req, res) => {

let obj = {

"p\_id": req.body.p\_id,

"uname": req.body.uname

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection:- ', err)

else {

let db = conn.db('miniprj')

db.collection('cart').deleteOne(obj, (err, result) => {

if (err)

res.json({ 'cartDelete': 'Error ' + err })

else {

if (result.deletedCount != 0) {

console.log(`Cart data fro ${obj.uname} deleted`)

res.json({ 'cartDelete': 'success' })

}

else {

console.log('Cart Data Not deleted')

res.json({ 'cartDelete': 'Record Not found' })

}

conn.close()

}

})

}

})

})

//Delete user

//to be done by participants

//export router

module.exports = router

Hosting the application

>npm init

1. create '.gitignore' file

>npx gitignore node

2. login to github.com and create repository

3. copy url

* url

4. initialyse local repository

>git init

5. add files to repository

>git add .

6. check status

>git status

7. commit

>git commit -m "initial Commit"

8. add to remote repository

>git remote add origin --Copied URL --

9. push code to repository

>git push -u origin master

Deploying nodejs on cyclic.sh

\*Login cyclic.sh with github

1. Click on deploy now

2. Select Link your own

3. Search and select required repository

4. Click on connect

5. Wait to deploy it

6. after getting message 'You're Live!'