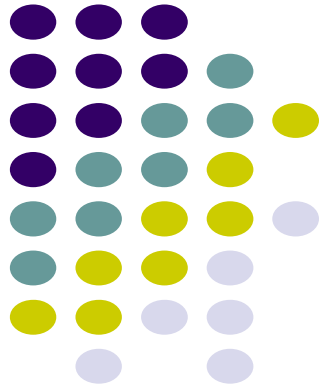
MongoDB

Dr. Arul Xavier V M



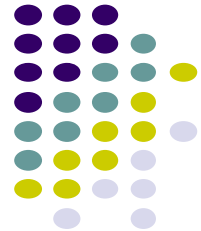
What is MongoDB

- MongoDB is a document database.
- It stores data in a type of JSON format called BSON(Binary JSON)
- A record in MongoDB is a document, which is a data structure composed of key value pairs similar to the structure of JSON objects.

Example MongoDB Document

```
{
        title: "Post Title 1",
        body: "Body of post.",
        category: "News",
        likes: 1,
        tags: ["news", "events"],
        date: Date()
}
```

How to install MongoDB



- MongoDB is a document database and can be installed locally or hosted in the cloud.
  - MongoDB Community Edition Server
    - https://www.mongodb.com/try/download/community
  - MongoDB Shell (mongosh)
    - https://www.mongodb.com/try/download/shell
  - MongoDB Compass (GUI Tool)
    - https://www.mongodb.com/try/download/compass

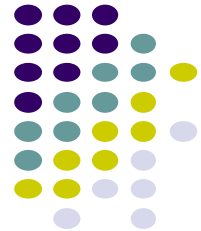
Connect to Mongo DB Server

- Open Mongo DB Shell (Mongosh shell)
  - mongodb://localhost/

MongoDB Query API

- The MongoDB Query API is the way you will interact with your data.
  - Create database
  - Create Collection
  - Insert Record
  - Find Record
  - Update Record
  - Delete Record

Create Database using mongosh



- Show all databases
  - show dbs
- Change or Create a Database
  - use database_name

```
test> show dbs
YourNewDatabase    8.00 KiB
admin             40.00 KiB
college            8.00 KiB
config           108.00 KiB
local             72.00 KiB
vmax              80.00 KiB
test> use college
switched to db college
college>
```

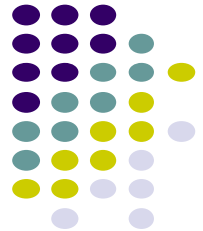
Remember: In MongoDB, a database is not actually created until it gets content!

Create a Collection

- Collection is similar to a table in a database, but the architecture of collection similar to JSON objects.
  - Create Collection using mongosh
    - You can create a collection using the createCollection() database method.
    - Display all the collections, show collections
    - Example,

```
mongosh mongodb://localhost/?directConnection=true&serverSelectionTimeoutMS=2000

college> db.createCollection('students')
{ ok: 1 }
college> show collections
students
college>
```

Delete Collection



- drop() function is used to delete the collection in mongodb.

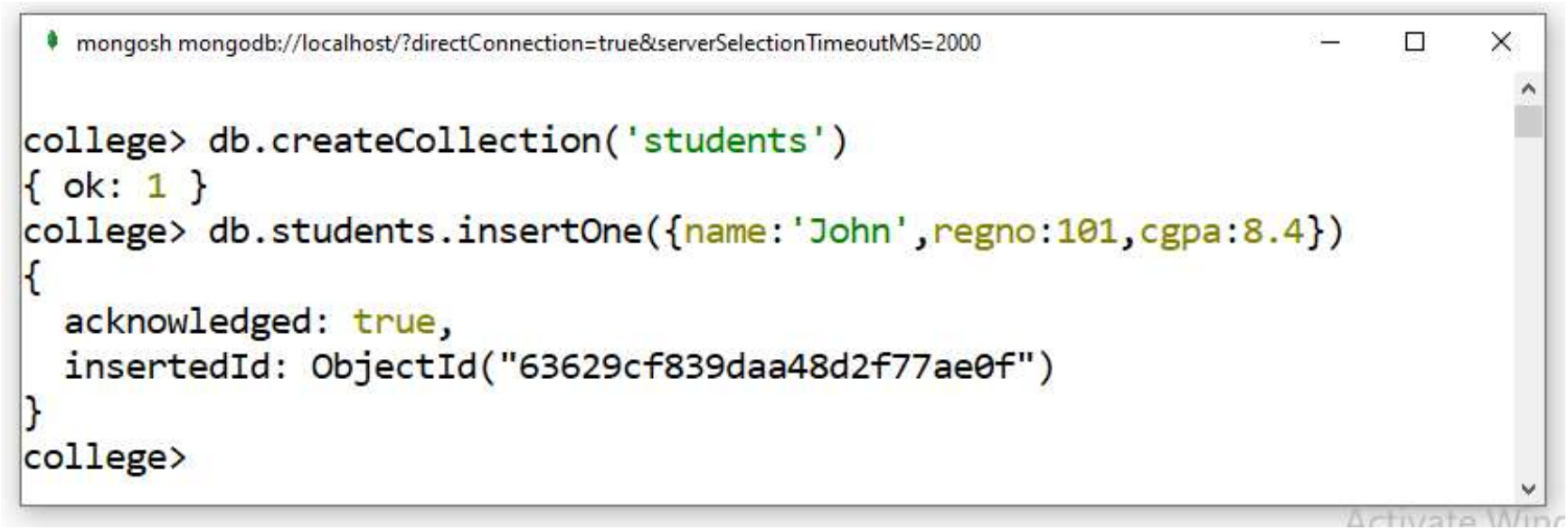
```
mongosh mongodb://localhost/?directConnection=true&serverSelectionTimeoutMS=2000

college> db.createCollection('students')
{ ok: 1 }
college> show collections
students
college> db.students.drop()
true
college> show collections

college>
```

Insert Documents

- Document is similar to a record of information about a particular object.
- There are 2 methods to insert documents into a MongoDB database.
  - insertOne()
    - To insert a single document, use the insertOne() method.
  - insertMany()
    - To insert multiple documents at once, use the insertMany() method.
    - This method inserts an array of objects into the database.

Insert Single Document

- insertOne(javasscript_object)
  - Example: To store information about one student such as name, regno, and cgpa



Insert Multiple Documents

- insertMany(jsarray) -
  - This method inserts an array of objects into the database.

```
mongosh mongodb://localhost/?directConnection=true&serverSelectionTimeoutMS=2000

college> db.students.insertMany([
...   {name:'John', regno:102, cgpa:7.1},
...   { name:'Vmax',regno:103, cgpa:6.7},
...   { name:'Suraj', regno:104, cgpa:8.5}
...   ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63629f8c39daa48d2f77ae15"),
    '1': ObjectId("63629f8c39daa48d2f77ae16"),
    '2': ObjectId("63629f8c39daa48d2f77ae17")
  }
}
college>
```

Read Documents

- There are 2 methods to find and select data from a MongoDB collection
  - find()
    - This method accepts a query object. If left empty, all documents will be returned.
  - findOne()
    - To select only one document, we can use the findOne() method.
    - This method accepts a query object. If left empty, it will return the first document it finds.

Read all documents

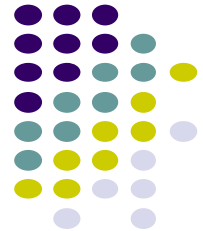
- find()

Output of find()

After inserting Multiple Documents

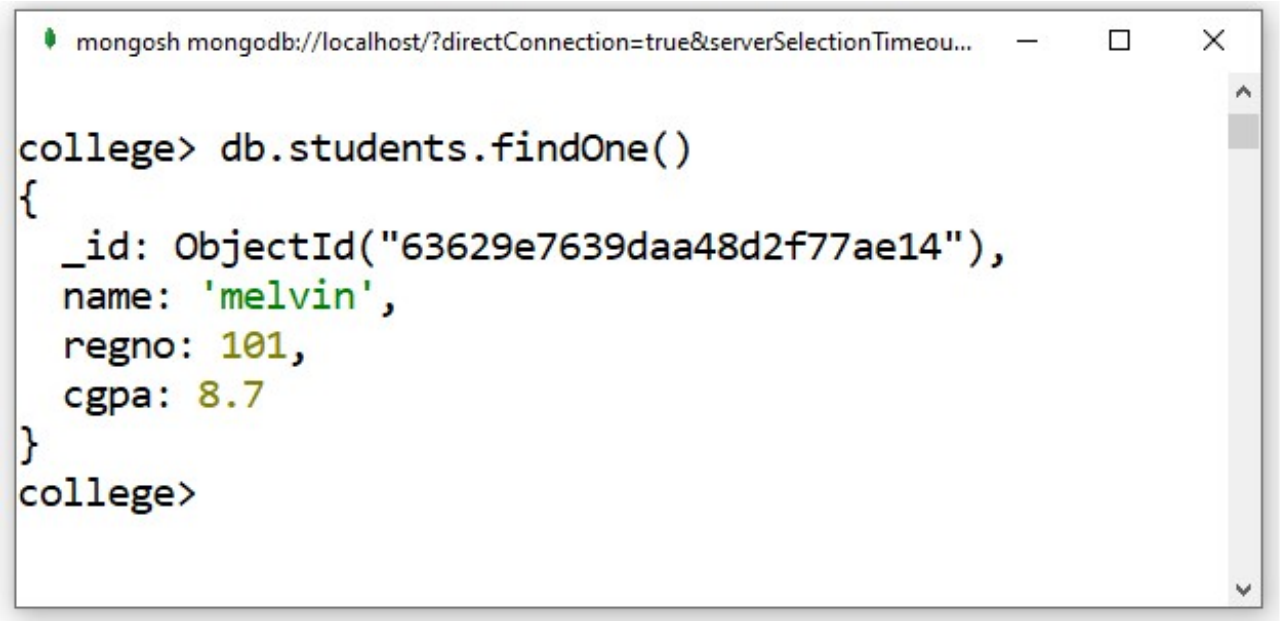
```
mongosh mongodb://localhost/?directConnection=true&serverSelectionTimeoutMS=2000

college> db.students.insertMany([
...   {name:'John', regno:102, cgpa:7.1},
...   { name:'Vmax',regno:103, cgpa:6.7},
...   { name:'Suraj', regno:104, cgpa:8.5}
...   ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63629f8c39daa48d2f77ae15"),
    '1': ObjectId("63629f8c39daa48d2f77ae16"),
    '2': ObjectId("63629f8c39daa48d2f77ae17")
  }
}
college>
```

```
college> db.students.find()
[
  {
    _id: ObjectId("63629e7639daa48d2f77ae14"),
    name: 'melvin',
    regno: 101,
    cgpa: 8.7
  },
  {
    _id: ObjectId("63629f8c39daa48d2f77ae15"),
    name: 'John',
    regno: 102,
    cgpa: 7.1
  },
  {
    _id: ObjectId("63629f8c39daa48d2f77ae16"),
    name: 'Vmax',
    regno: 103,
    cgpa: 6.7
  },
  {
    _id: ObjectId("63629f8c39daa48d2f77ae17"),
    name: 'Suraj',
    regno: 104,
    cgpa: 8.5
  }
]
college>
```

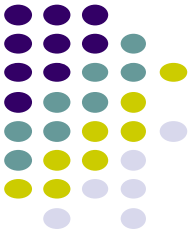


Reading Single Document

- findOne() function used to find single document, it returns first document by default.
  - Query object can be used to find a particular record.



Read Modifiers



- sort() method
  - Sort the results of a find by the given fields
  - db.Collection_Name.find().sort({filed_name:1 or -1})
    - 1 => Ascending Order , -1 => Descending order

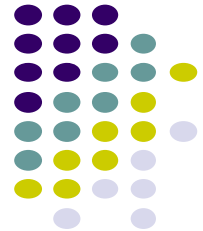
Read Modifiers

- limit() method
  - Only return a set number of documents
  - db.Collection_Name.find().limit(N)
    - N – number of documents to be fetched

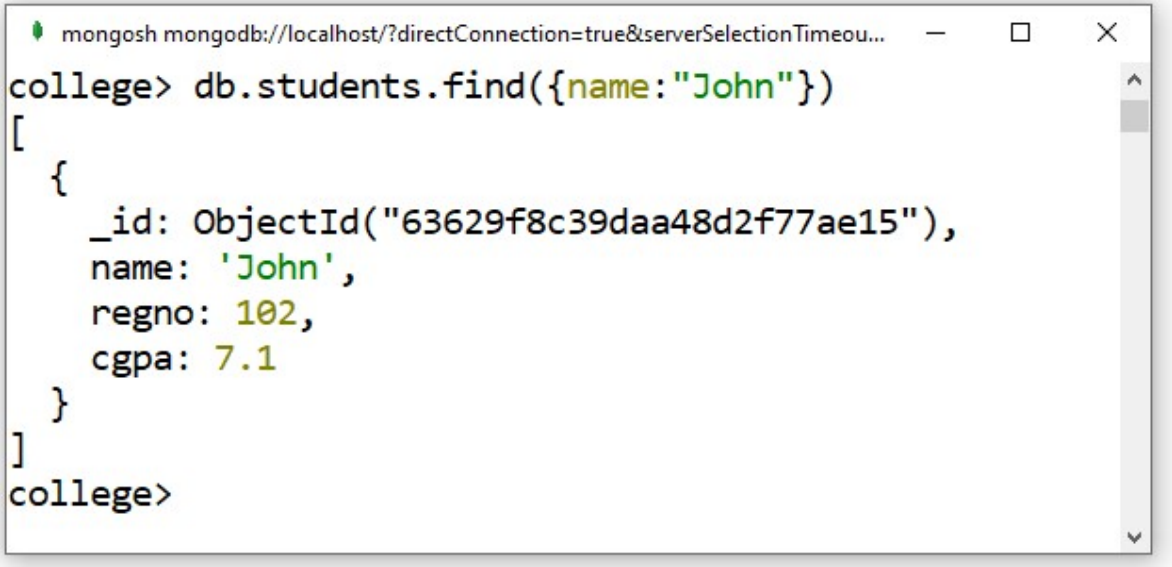
Read Modifiers

- skip() method
  - Skip a set number of documents from the beginning
  - db.Collection_Name.find().skip(N)
    - N – number of documents to be skipped

Querying Data or filter data



- To query, or filter, data we can include a query in our find() or findOne() methods.
- Example
  - To find the record whose name is ‘john’



Projection

- Both find methods accept a second parameter called projection.
- This parameter is an object that describes which fields to include in the results.
  - We use a 1 to include a field and 0 to exclude a field.

```
mongosh mongodb://localhost/?directConnection=true&serverSelectionTimeoutMS=2000

college> db.students.find({name:"John"},{cgpa:1})
[ { _id: ObjectId("63629f8c39daa48d2f77ae15"), cgpa: 7.1 } ]
college>

college> db.students.find({name:"John"},{_id:0,cgpa:1})
[ { cgpa: 7.1 } ]
college> db.students.find({name:"John"},{_id:0,cgpa:1,regno:1})
[ { regno: 102, cgpa: 7.1 } ]
college>

college> db.students.find({name:"John"},{_id:0,cgpa:0,regno:1})
MongoServerError: Cannot do inclusion on field regno in exclusion projection
college>
```

Error if we try to specify both 0 and 1 in the same object, except _id field

Query Operators

- $eq: Values are equal
- $ne: Values are not equal
- $gt: Value is greater than another value
- $gte: Value is greater than or equal to another value
- $lt: Value is less than another value
- $lte: Value is less than or equal to another value
- $in: Value is matched within an array

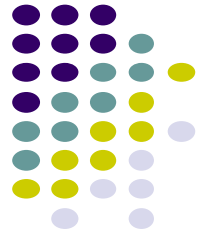
Example

```
>_MONGOSH

> db.students.find({cgpa:{$gte:8.4}})
< { _id: ObjectId("63634f0b86d0c192ac8d5a2a"),
    name: 'arun',
    regno: 102,
    cgpa: 8.8 }
  { _id: ObjectId("6363506086d0c192ac8d5a2b"),
    name: 'raj',
    regno: 103,
    cgpa: 9.1 }
college>
```

Update Documents

- To update an existing document we can use the following methods.
  - updateOne()
  - updateMany()
- The first parameter is a query object to define which document or documents should be updated.
- The second parameter is an object defining the updated data.
  - To update we need to use $set query operator

updateOne()



- The updateOne() method will update the first document that is found matching the provided query.
  - Example: to update the cgpa of a student whose name is ‘raj’

```
>_MONGOSH
> db.students.updateOne({name:"raj"},{$set:{cgpa:9.0}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.students.find({name:"raj"})
< { _id: ObjectId("6363506086d0c192ac8d5a2b"),
    name: 'raj',
    regno: 103,
    cgpa: 9 }
college>
```

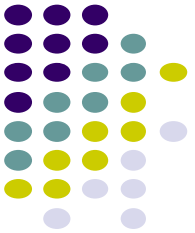
Insert if Document not found

- You can insert a new document if it is not exists via update functionality.
  - {upsert:true}

Here, the document “joy” is not exists, it will be inserted

```
> db.students.updateOne({name:"joy"}, {$set:{name:"joy",regno:450,cgpa:9.5}}, {upsert:true})
< { acknowledged: true,
    insertedId: ObjectId("6363f2328b13f3d112be5d84"),
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 1 }
> db.students.find({name:"joy"})
< { _id: ObjectId("6363f2328b13f3d112be5d84"),
    name: 'joy',
    cgpa: 9.5,
    regno: 450 }
college>
```

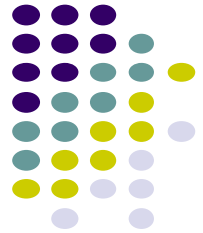
updateMany()



- You can multiple documents that matches the query selector.
  - To select all documents, use empty { } as query selector

```
> db.students.updateMany({},{$set:{gender:'Male'}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 7,
    modifiedCount: 7,
    upsertedCount: 0 }
```

updateMany()



- To update all “Male” students cgpa to 8.5

```
> db.students.updateMany({gender:"Male"},{$set:{cgpa:8.5}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 6,
    modifiedCount: 6,
    upsertedCount: 0 }
college>
```

Delete Documents

- We can delete documents by using the methods
  - deleteOne()
  - deleteMany().

deleteOne()

- The deleteOne() method will delete the first document that matches the query provided.

```
>_MONGOSH

> db.students.deleteOne({name:"vincy"})
< { acknowledged: true, deletedCount: 1 }
> db.students.find({name:"vincy"})
<
college>
```

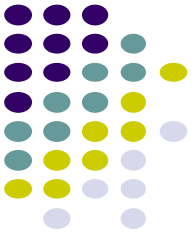
deleteMany()

- The deleteMany() method will delete all documents that match the query provided.

```
college> db.students.deleteMany({gender:"Male"})
```

This will delete all student documents whose gender is “Male”

mongoose module



- Mongoose enables you to interact with MongoDB from Node JS applications.
- Mongoose is an abstraction over the native MongoDB driver.
- The standard Node module used for database operation is Mongoose.
- Mongoose exposes all the MongoDB features in an easy way and JavaScript-friendly interface

Create Database Connection

```
const mongoose = require('mongoose');
mongoose.connect("mongodb://localhost/company", function(){
    console.log('database connected');
})
```

Mongoose key concepts

- Schema
  - Represents a database or collection in MongoDB
- Model
  - Represents a document or record in MongoDB collection.
- Query
  - Functions to manipulate data in collection.

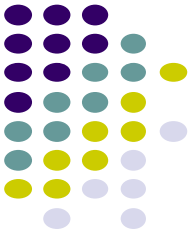
Schema

- A schema maps to a MongoDB document or collection and is a set of rules and instructions for creating models.
- Few SchemaTypes are
  - String
  - Number
  - Date
  - Boolean
  - Array

Example Mongoose Schema

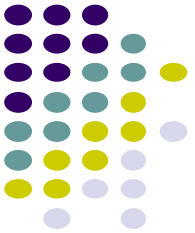
```
const employeeSchema = new mongoose.Schema({
    name:String,
    empid:Number,
    salary:Number
})
```

Create Model



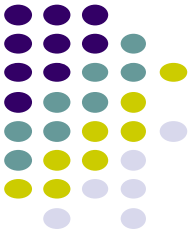
```
const Employee = new mongoose.model('Employees',employeeSchema);
```

Insert Document



```
try {
    const emp = Employee.create({name:"raj",empid:423,salary:500000})
    console.log("record saved..")
}catch(error){
   console.log(error)
}
```

Find document

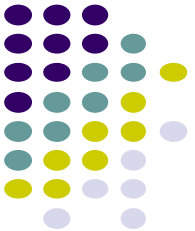


```
const users = Employee.find({}, function(err, docs){
    console.log(docs);
})
```

Iterate after finding

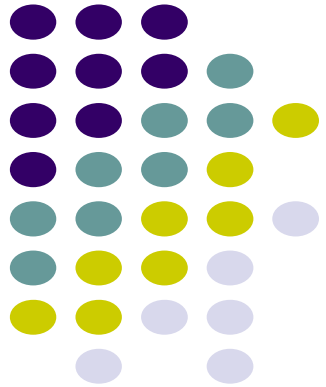
```
const users = Employee.find({},{_id:0,__v: 0},function(err, docs){
    docs.forEach(element => {
        console.log('Name: ' + element.name);
    });
})
```

Update Document

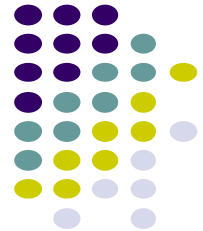


```
const user=Employee.updateOne({name:"vmax"}, {$set: {salary: 400000}}, function(err){
    console.log('updated' + user)
})
```

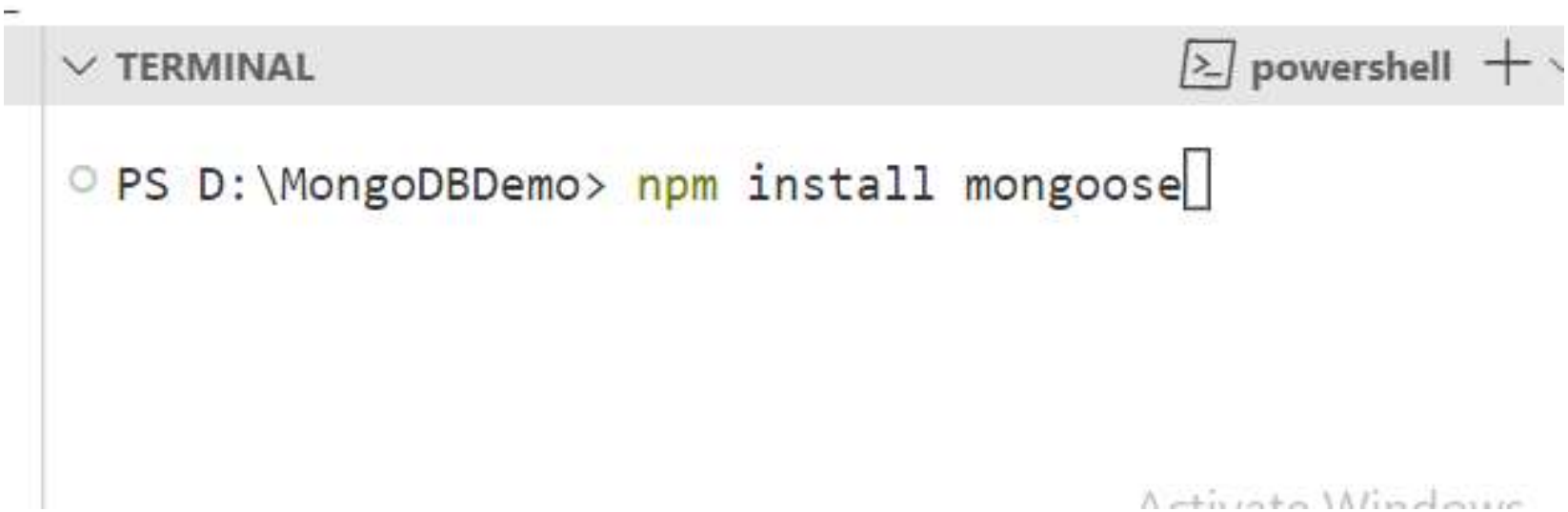
Node JS Program to save data in Database



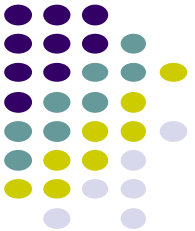
Install the mongoose module



PS D:\MongoDBDemo> npm install mongoose



Create a HTML Form

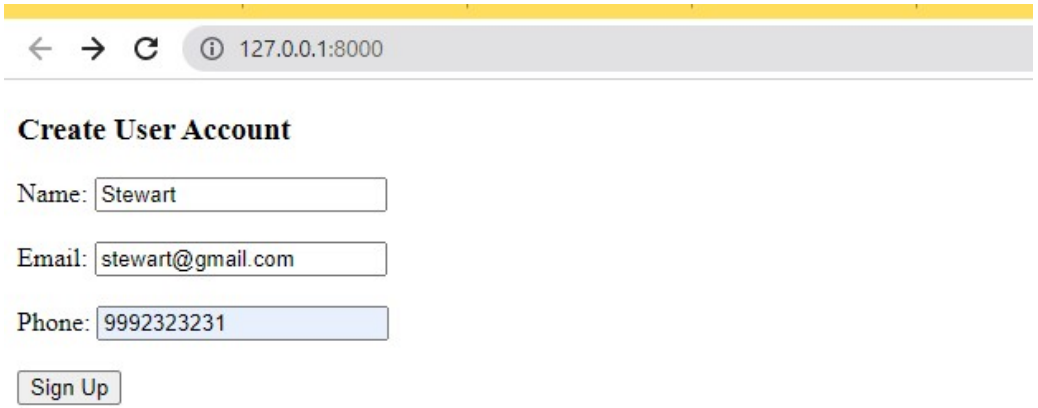


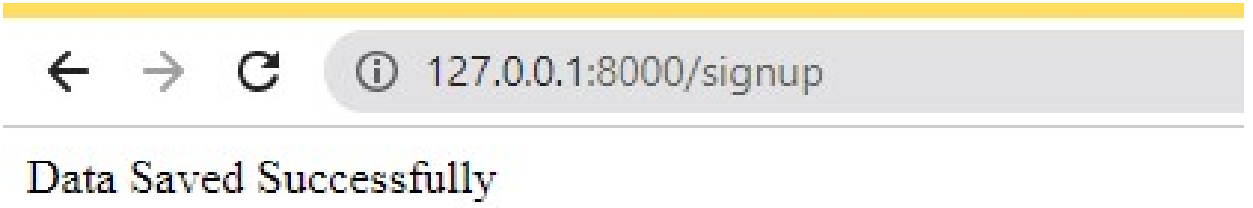
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign Up Page</title>
</head>
<body>
    <form action="/signup" method="POST">
        <h3>Create User Account</h3>
        Name: <input type="text" name="name" required><br><br>
        Email: <input type="email" name="email" required><br><br>
        Phone: <input type="text" name="phone" required><br><br>
        <button type="submit">Sign Up</button>
    </form>
    <br><br>
    </body>
</html>
```

```
const http = require('http');
const fs = require('fs');

//To include mongoose module in node js program
const mongoose = require('mongoose');

//Connecting to the mongodb database
mongoose.connect('mongodb://127.0.0.1:27017/college')
        .then(function(){
            console.log('DB Connected')
        })

//Defining the Structure of mongodb document
const studentSchema = new mongoose.Schema({name:String, email:String,phone:String});

//Create collection model
const studentmodel = mongoose.model('students',studentSchema);
```

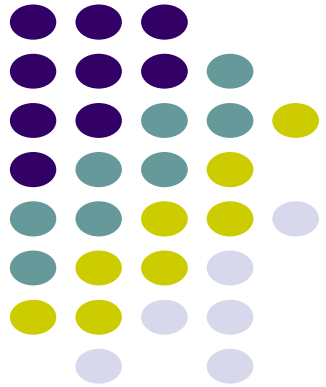
```
const server = http.createServer(function(req,res){
      if(req.url === '/'){
         res.writeHead('200',{'Content-Type':'text/html'});
         fs.createReadStream('signup.html').pipe(res);
      }
      else if(req.url=== '/signup' && req.method === 'POST'){
          var rawdata = '';
          req.on('data',function(data){
              rawdata += data;
          })
          req.on('end',function(){
              var formdata = new URLSearchParams(rawdata);
              res.writeHead('200',{'Content-Type':'text/html'});
              studentmodel.create({name:formdata.get('name'),
                                   email:formdata.get('email'),
                                   phone:formdata.get('phone')
                                  })
              res.write('Data Saved Successfully');
              res.end();
          })
      }
  );
```

```
server.listen('8000',function(){
    console.log('Server started at port
http://127.0.0.1:8000');
})
```


127.0.0.1:8000
Create User Account
Name: Stewart
Email: stewart@gmail.com
Phone: 9992323231
Sign Up


127.0.0.1:8000/signup
Data Saved Successfully

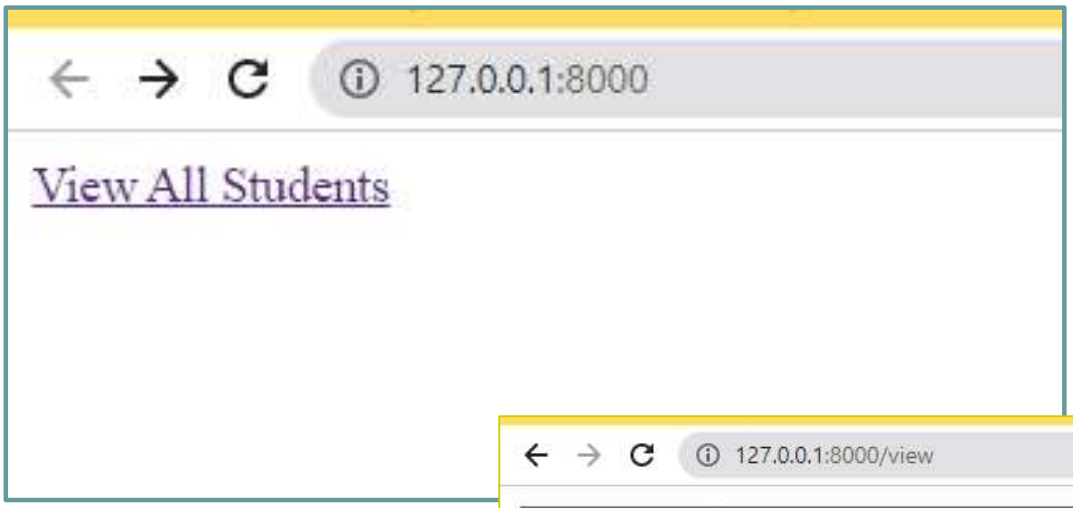
Node JS Program to Display all data from Database



Node JS program display all data

```
const http = require('http');
const fs = require('fs');

//To include mongoose module in node js program
const mongoose = require('mongoose');

//Connecting to the mongodb database
mongoose.connect('mongodb://127.0.0.1:27017/college')
        .then(function(){
            console.log('DB Connected')
        })

//Defining the Structure of mongodb document
const studentSchema = new mongoose.Schema({name:String, email:String,phone:String});

//Create collection model
const studentmodel = mongoose.model('students',studentSchema);
```

```
const server = http.createServer(function(req,res){
          res.writeHead('200',{'Content-Type':'text/html'});
            //To fetch all data from mongodb database collection
          studentmodel.find().then(function(students){
                res.write("<table border=1 cellspacing=0 width=400>");
                res.write("<tr><th>Name</th><th>Email</th><th>Phone</th></tr>");
                students.forEach(student=>{
                    res.write("<tr>");
                    res.write("<td>"+student.name+"</td>");
                    res.write("<td>"+student.email+"</td>");
                    res.write("<td>"+student.phone+"</td>");
                    res.write("</tr>");
                })
                res.end();
            })
})
server.listen('8000',function(){
    console.log('Server started at port http://127.0.0.1:8000');
})
```



Name	Email	Phone
John	john@gmail.com	8239232323
Raj	raj@gmail.com	9923239232
vmax	vmax@gmail.com	892329323
reban	reban@gmail.com	8923023232
Stewart	stewart@gmail.com	9992323231