7.c

Course Name: Express.js

Module Name: Connecting to MongoDB with Mongoose, Validation Types and Defaults

Write a Mongoose schema to connect with MongoDB.

# Program:

**Connecting mongoose to mongo db:**

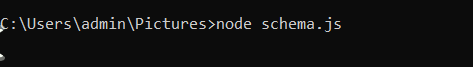
var mongoose = require("mongoose");

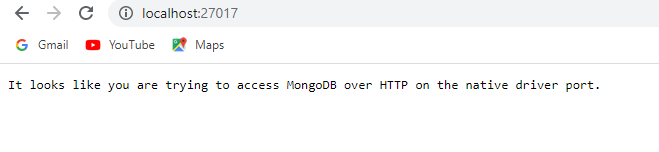
var dbHost = "mongodb://localhost:27017/test";

mongoose.connect(dbHost);

# Output:

Its connected to the mongodb





# Program:

**Creating a schema**

**Program:**

var mongoose = require("mongoose");

var dbHost = "mongodb://localhost:27017/test";

mongoose.connect(dbHost);

//Create a schema for Book

var bookSchema = mongoose.Schema({

name: String,

//Also creating index on field isbn

isbn: {type: String, index: true},

author: String,

pages: Number

});

# Output:

**It shows running**



7.d

Course Name: Express.js

Module Name: Models Write a program to wrap the Schema into a Model object.

Program:

**Creating an model**

//Create a Model by using the schema defined above

var Book = mongoose.model("Book", bookSchema, "mongoose\_demo");

var db = mongoose.connection;

//this tells that mongoose is connected to mongodb

db.on("error", console.error.bind(console, "connection error:"));

db.once("open", function(){

console.log("Connected to DB");

});

# Output:



**8.a**

**Course Name: Express.js**

**Module Name: CRUD Operations Write a program to perform various CRUD (Create-Read-Update-Delete) operations using Mongoose library functions.**

# Program:

var mongoose = require("mongoose");

var dbHost = "mongodb://localhost:27017/test";

mongoose.connect(dbHost);

//Create a schema for Book

var bookSchema = mongoose.Schema({

name: String,

//Also creating index on field isbn

isbn: {type: String, index: true},

author: String,

pages: Number

});

//Create a Model by using the schema defined above

//Optionally one can provide the name of collection where the instances

//of this model get stored. In this case it is "mongoose\_demo". Skipping

//this value defaults the name of the collection to plural of model name i.e books.

var Book = mongoose.model("Book", bookSchema, "mongoose\_demo");

var db = mongoose.connection;

db.on("error", console.error.bind(console, "connection error:"));

db.once("open", function(){

console.log("Connected to DB");

//Instantiating the Model – An instance of Model represents a mongodb document

var book1 = new Book({

name:"Mongoose Demo 1",

isbn: "MNG123",

author: "Author1, Author2",

pages: 123

});

//Saving the model instance to the DB

book1.save(function(err){

if ( err )

throw err;

console.log("Book Saved Successfully");

});

var book2 = new Book({

name:"Mongoose Demo 2",

isbn: "MNG124",

author: "Author2, Author3",

pages: 90

});

book2.save(function(err){

if ( err )

throw err;

console.log("Book Saved Successfully");

deleteBook();

});

var book3 = new Book({

name:"Mongoose Demo 3",

isbn: "MNG125",

author: "Author2, Author4",

pages: 80

});

book3.save(function(err){

if ( err )

throw err;

console.log("Book Saved Successfully");

queryBooks();

updateBook();

});

});

var queryBooks = function(){

//Now querying those books which have less than 100 pages

//Find API takes in a query condition, attributes in the document to be projected,

//callback to be executed when the data is available.

Book.find({pages : {$lt:100}}, "name isbn author pages", function(err, result){

if ( err )

throw err;

console.log("Find Operations: " + result);

});

}

var updateBook = function(){

/\*

Find the book to be updated using the condition and then execute the update

passed to the API as the second argument

\*/

Book.update({isbn : {$eq: "MNG125"}}, {$set: {name: "Mongoose Demo 3.1"}}, function(err, result){

console.log("Updated successfully");

console.log(result);

});

}

var deleteBook = function(){

/\*

When callback is not passed, the action is not invoked on the collection

until the exec() method is called.

In this case I am not passing the callback and instead executing the action

by invoking the exec() method.

\*/

Book.remove({isbn:{$eq: "MNG124"}}).exec();

}

Output:

