PROJECT 4

Task

In this assignment you are going to implement a simple Book Register web form using MEAN stack.

Step 1: Install NodeJs

`Node.js` is a JavaScript runtime built on Chrome's V8 JavaScript engine. `Node.js` is used in this tutorial to set up the Express routes and AngularJS controllers.

Update ubuntu

```
sudo apt update
```

Upgrade ubuntu

```
sudo apt upgrade
```

Add certificates

```
sudo apt -y install curl dirmngr apt-transport-https lsb-release ca-certificates
```

```
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

Install NodeJS

```
sudo apt install -y nodejs
```

Step 2: Install MongoDB

MongoDB stores data in flexible, `JSON-like` documents. Fields in a database can vary from document to document and data structure can be changed over time. For our example application, we are adding book records to MongoDB that contain book name, isbn number, author, and number of pages.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A14518585931BC711F9BA15703C6
```

Install MongoDB

```
sudo apt install -y mongodb
```

Start The server

```
sudo service mongodb start
```

Verify that the service is up and running

```
sudo systemctl status mongodb
```

```
ubuntu@ip-172-31-32-234:~$ sudo systemctl status mongodb
● mongodb.service - An object/document-oriented database
     Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-06-05 13:21:41 UTC; 1min 20s ago
       Docs: man:mongod(1)
   Main PID: 20776 (mongod)
      Tasks: 23 (limit: 1111)
     Memory: 43.2M
     CGroup: /system.slice/mongodb.service
             └─20776 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf

Jun 05 13:21:41 ip-172-31-32-234 systemd[1]: Started An object/document-oriented database.
ubuntu@ip-172-31-32-234:~$
```

Install npm – Node package manager.

```
sudo apt install -y npm
```

Install body-parser package

We need 'body-parser' package to help us process JSON files passed in requests to the server.

```
sudo npm install body-parser
```

Create a folder named 'Books'

```
mkdir Books && cd Books
```

In the Books directory, Initialize npm project

```
npm init
```

Add a file to it named `server.js`

```
vi server.js
```

Copy and past as below

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(express.static(__dirname + '/public'));
app.use(bodyParser.json());
require('./apps/routes')(app);
app.set('port', 3300);
app.listen(app.get('port'), function() {
    console.log('Server up: http://localhost:' + app.get('port'));
});
```

Step 3: Install Express and set up routes to the server

```
sudo npm install express mongoose
```

In ‘Books’ folder, create a folder named `apps`

```
Sudo mkdir apps && cd apps
vi routes.js
```

Copy and paste the code below into routes.js

```
const Book = require('./models/book');

module.exports = function(app){
  app.get('/book', function(req, res){
    Book.find({}).then(result => {
      res.json(result);
    }).catch(err => {
      console.error(err);
      res.status(500).send('An error occurred while retrieving books');
    });
  });

  app.post('/book', function(req, res){
    const book = new Book({
      name: req.body.name,
      isbn: req.body.isbn,
      author: req.body.author,
      pages: req.body.pages
    });
    book.save().then(result => {
      res.json({
        message: "Successfully added book",
        book: result
      });
    }).catch(err => {
      console.error(err);
      res.status(500).send('An error occurred while saving the book');
    });
  });

  app.delete("/book/:isbn", function(req, res){
    Book.findOneAndRemove(req.query).then(result => {
      res.json({
        message: "Successfully deleted the book",
        book: result
      });
    }).catch(err => {
      console.error(err);
      res.status(500).send('An error occurred while deleting the
book');
```

```
    });
  });

  const path = require('path');
  app.get('*', function(req, res){
    res.sendFile(path.join(__dirname, 'public', 'index.html'));
  });
};
```

In the ‘apps’ folder, create a folder named `models`

```
mkdir models && cd models
```

Create a file named `book.js`

```
vi book.js
```

Copy and paste the code below into ‘book.js’

```
var mongoose = require('mongoose');
var dbHost = 'mongodb://localhost:27017/test';
mongoose.connect(dbHost);
mongoose.connection;
mongoose.set('debug', true);
var bookSchema = mongoose.Schema( {
  name: String,
  isbn: {type: String, index: true},
  author: String,
  pages: Number
});
var Book = mongoose.model('Book', bookSchema);
module.exports = mongoose.model('Book', bookSchema);
```

Step 4 – Access the routes with AngularJS

AngularJS provides a web framework for creating dynamic views in your web applications. In this tutorial, we use AngularJS to connect our web page with Express and perform actions on our book register.

Change the directory back to ‘Books’

Create a folder named `public`

```
mkdir public && cd public
```

Add a file named `script.js`

```
vi script.js
```

Copy and paste the Code below (controller configuration defined) into the script.js file

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
  $http( {
    method: 'GET',
    url: '/book'
  }).then(function successCallback(response) {
    $scope.books = response.data;
  }, function errorCallback(response) {
    console.log('Error: ' + response);
  });
  $scope.del_book = function(book) {
    $http( {
      method: 'DELETE',
      url: '/book/:isbn',
      params: {'isbn': book.isbn}
    }).then(function successCallback(response) {
      console.log(response);
    }, function errorCallback(response) {
      console.log('Error: ' + response);
    });
  };
  $scope.add_book = function() {
    var body = '{ "name": "' + $scope.Name +
    '", "isbn": "' + $scope.Isbn +
    '", "author": "' + $scope.Author +
    '", "pages": "' + $scope.Pages + '" }';
    $http({
      method: 'POST',
      url: '/book',
      data: body
    }).then(function successCallback(response) {
      console.log(response);
    }, function errorCallback(response) {
      console.log('Error: ' + response);
    });
  };
});
```

In `public` folder, create a file named `index.html`;

```
vi index.html
```

Cpoy and paste the code below into `index.html` file.

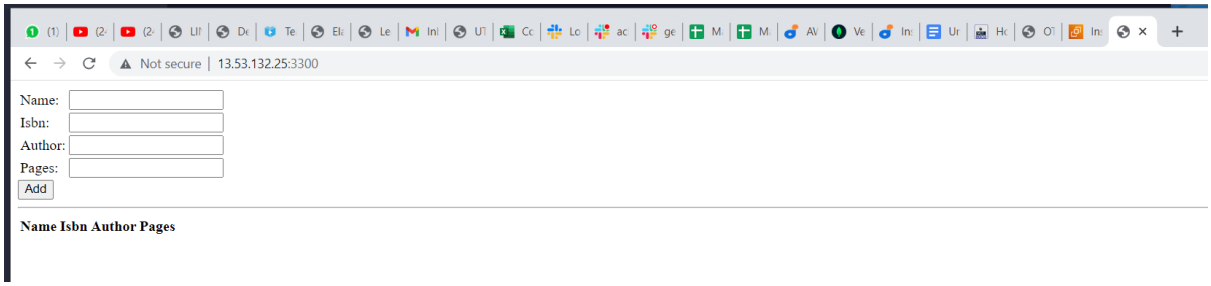
```
<!doctype html>
<html ng-app="myApp" ng-controller="myCtrl">
```

```
  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.
js"></script>
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <table>
        <tr>
          <td>Name:</td>
          <td><input type="text" ng-model="Name"></td>
        </tr>
        <tr>
          <td>Isbn:</td>
          <td><input type="text" ng-model="Isbn"></td>
        </tr>
        <tr>
          <td>Author:</td>
          <td><input type="text" ng-model="Author"></td>
        </tr>
        <tr>
          <td>Pages:</td>
          <td><input type="number" ng-model="Pages"></td>
        </tr>
      </table>
      <button ng-click="add_book()">Add</button>
    </div>
    <hr>
    <div>
      <table>
        <tr>
          <th>Name</th>
          <th>Isbn</th>
          <th>Author</th>
          <th>Pages</th>

        </tr>
        <tr ng-repeat="book in books">
          <td>{{book.name}}</td>
          <td>{{book.isbn}}</td>
          <td>{{book.author}}</td>
          <td>{{book.pages}}</td>

          <td><input type="button" value="Delete" data-ng-
click="del_book(book)"></td>
        </tr>
      </table>
    </div>
  </body>
</html>
```

Change the directory back up to `Books`

Start the server by running this command:
`node server.js`

The server is now up and running, we can connect it via port 3300. You can launch a separate Putty or SSH console to test what `curl` command returns locally.



Project 4 finished…..Thanks