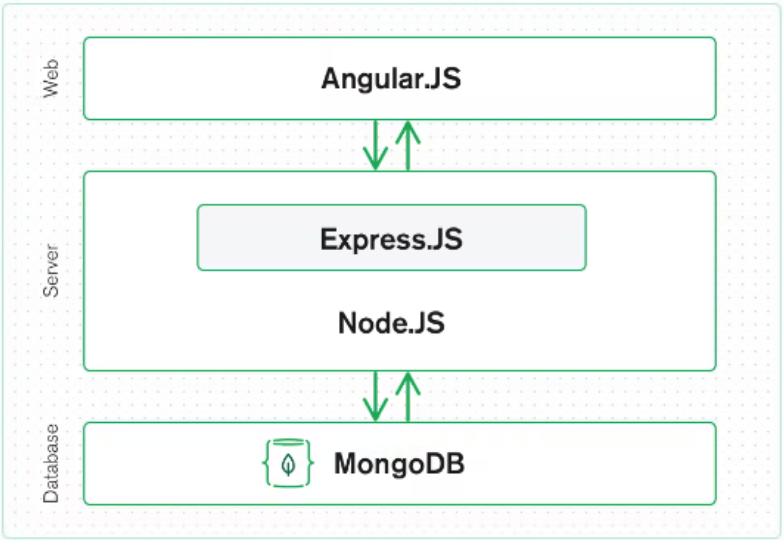
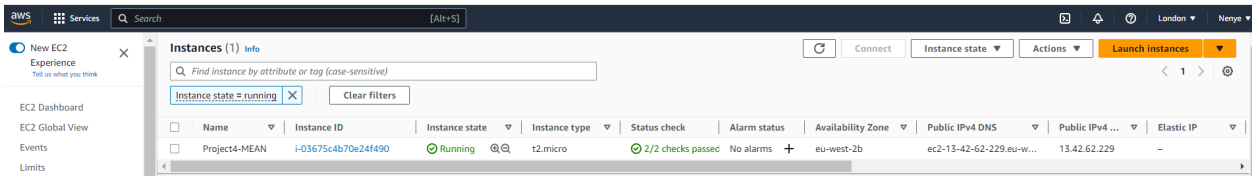
PROJECT 4: MEAN STACK IMPLEMENTATION

MEAN is a web application development framework based on JavaScript. It combines some of the contemporary web technologies such as MongoDB as a NoSQL database system, Express as a web server framework for Node, AngularJS as client-side scripting and Node as a server platform which is then used to form a stack. The integration between the components of the stack is logical and simple because all of the components are based on JavaScript and JSON. Each component of the MEAN stack is open source, giving everyone a wide range of free opportunities to use it. Below is a diagram of the MEAN Stack architecture and in this project, I will be using the MEAN stack to implement a Book Register web form.



Launch an EC2 Instance

A step-by-step process of how to do this has been documented in Project 1. My instance is running and will now need to connect to an ssh client.



Installing Node.js

Node.js is an open-source, cross-platform runtime environment for JavaScript designed to build scalable network applications that can manage many connections at once. To commence this installation, it is important to run the below commands to update and upgrade Ubuntu.

```
ubuntu@ip-172-31-34-6:~$ sudo apt upgrade -y
```

```
ubuntu@ip-172-31-34-6:~$ sudo apt update -y
```

To add a certificate and then install the actual node.js I ran:

```
ubuntu@ip-172-31-34-6:~$ 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 -
```

```
ubuntu@ip-172-31-34-6:~$ sudo apt install -y nodejs
```

Now this has been completed successfully, it's time to move to the next step which is to get the database ready.

Installing MongoDB

In contrast to relational databases, which use rows and columns to store data, MongoDB is a flexible, and scalable NoSQL document store model that stores data in a JSON format called BJSON, which stands for Binary JSON. The Book Register web form will contain the book name, ISBN, author, and the number of pages and these records will be stored in the MongoDB. To import keys as per the official installation documentation for MongoDB, I ran these commands.

```
ubuntu@ip-172-31-34-6:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A14518585931BC711F9BA15703C6
```

```
ubuntu@ip-172-31-34-6:~$ echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/
mongodb-org-3.4.list
```

Once completed, I installed MongoDB and then started the service.

```
ubuntu@ip-172-31-34-6:~$ sudo apt install -y mongodb
```

```
ubuntu@ip-172-31-34-6:~$ sudo service mongodb start
```

To ensure that the service is running, I ran the sudo systemctl status Mongodb. The output should appear as active (running) which shows that the Nginx service is up and available.

```
ubuntu@ip-172-31-34-6:~$ 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 Sat 2023-06-03 17:29:20 UTC; 30min ago
       Docs: man:mongod(1)
   Main PID: 458 (mongod)
      Tasks: 23 (limit: 1141)
     Memory: 53.8M
     CGroup: /system.slice/mongodb.service
             └─458 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf

Jun 03 17:29:20 ip-172-31-34-6 systemd[1]: Started An object/document-oriented database.
```

I will need to install npm, the default package manager for the JavaScript runtime environment Node.js.

```
ubuntu@ip-172-31-34-6:~$ sudo apt install -y npm
```

An HTTP request body is parsed by a body parser, installing this will help especially when I need more information about the requests sent to the server rather than just the URL being accessed.

```
ubuntu@ip-172-31-34-6:~$ sudo npm install body-parser
```

Now it's time I created a new directory named books, cd into the new directory and then initialize the npm project.

```
ubuntu@ip-172-31-34-6:~$ mkdir Books && cd Books
```

```
ubuntu@ip-172-31-34-6:~/Books$ npm init
```

To add a server.js file and edit it, I used the vi command.

```
ubuntu@ip-172-31-34-6:~/Books$ vi server.js
```

Now copy and paste the default settings for the webserver, click on esc, type ‘i’ to insert, ‘:w’ to save and then :qa to exit or just use :wq! To save and exit.

```
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'));
});
```

Installing Express

Express.js will be used to transfer book data to and from the MongoDB database. It is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. A connection will need to be made between MongoDB and the Node.js JavaScript runtime environment through the object-oriented JavaScript programming framework called Mongoose. The database's schema will need to be established to hold the data from our book register.

```
ubuntu@ip-172-31-34-6:~/Books$ sudo npm install express mongoose
```

I created a new directory in the books folder named Apps, I cd into the new directory.

```
ubuntu@ip-172-31-34-6:~/Books$ mkdir apps && cd apps
```

Then I created a file named routes.js and edited it using the vi command **vi routes.js**. Now copy and paste the default settings and save.

```
    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'));
});
};
```

I made a folder called models while still in the apps folder, changed the directory afterwards, and then created a book.js file.

```
ubuntu@ip-172-31-34-6:~/Books/apps$ mkdir models && cd models
```

```
ubuntu@ip-172-31-34-6:~/Books/apps/models$ vi book.js
```

I pasted the below before saving the book.js file.

```
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);
```

Accessing the routes with AngularJS

AngularJS is an open-source JavaScript-based web framework for developing single-page applications. It will be used to connect our web page with Express to perform actions in the book register. I will need to go back to the Books directory to make any changes by running.

```
ubuntu@ip-172-31-34-6:~/Books/apps/models$ cd ../..
```

Then create a public directory, cd into it and then use vi to create and edit the script.js file.

```
ubuntu@ip-172-31-34-6:~/Books$ mkdir public && cd public
```

```
ubuntu@ip-172-31-34-6:~/Books/public$ vi script.js
```

I pasted the below controller configuration defined code before saving 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 this same public folder, I created another file named index.html using the vi index.html command, pasted and saved the below

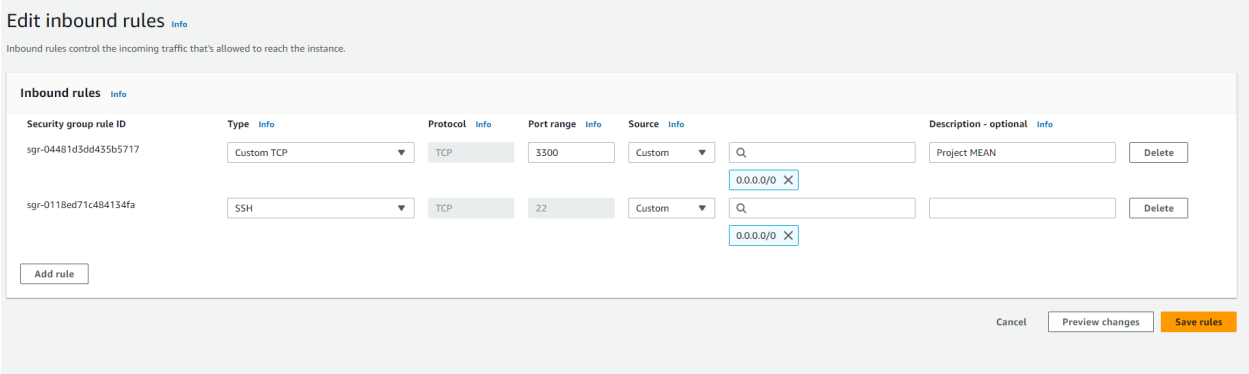
```
<!doctype html>
<html ng-app="myApp" ng-controller="myCtrl">
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <table>
        <tr>
          <td>Name:</td>
          <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>
```

Once this has been completed, I changed the directory back to books and started the server. This will give you a server-up output if everything worked well as it should.

```
ubuntu@ip-172-31-34-6:~/Books$ node server.js
Server up: http://localhost:3300
```

The next step now will be to open our TCP Port 3300 on our EC2 configuration to be able to establish a connection and exchange streams of data. Refer to Project 1 for steps to achieve

this.



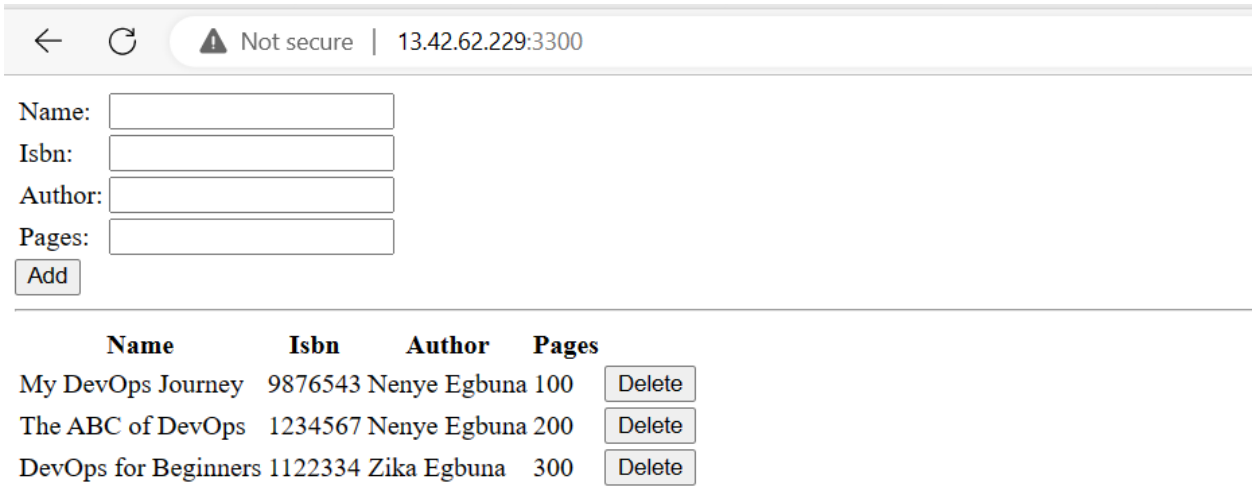
To confirm if the server is running by opening a new ssh and running

```
ubuntu@ip-172-31-34-6:~$ curl -s http://localhost:3300
```

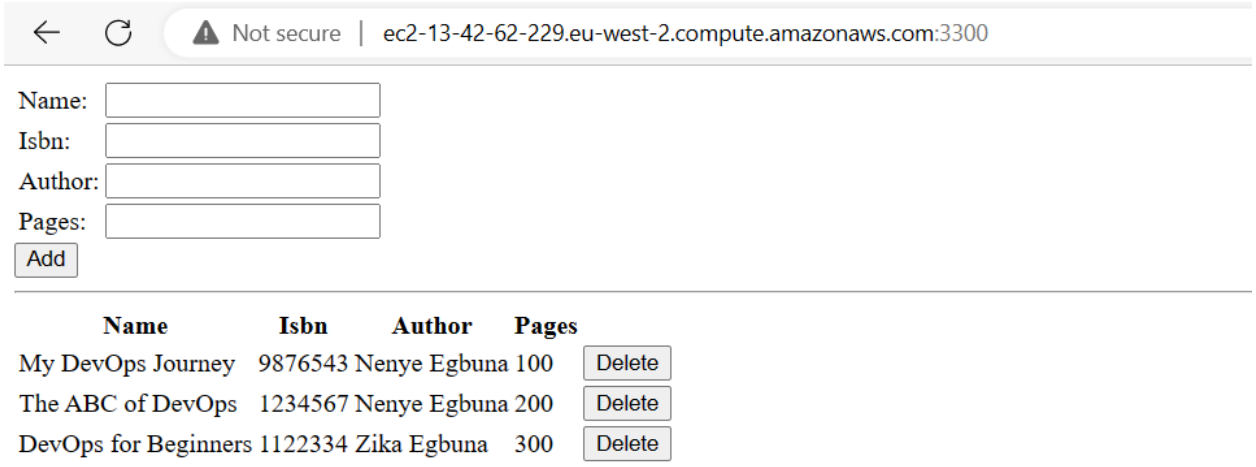
Results like this will let you know it is connected as it returned an HTML page, which is hardly readable from the terminal, but we can also try and access it from the Internet using the public IP or DNS.

```
ubuntu@ip-172-31-34-6:~$ curl -s http://localhost:3300
<!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>
```

Web Book Register Application using the Public IP address in my browser.



Web Book Register Application using the Public DNS in my browser.



You should get my books, I am officially a DevOps author! 😁

REFERENCES

Karanjit, A. (2016) *St. Cloud State University Therepository at St. Cloud State*. Available at: https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1039&context=csit_etds (Accessed: 30 May 2023).

What Is the MEAN Stack? (2023) MongoDB. Available at: https://www.mongodb.com/mean-stack/ (Accessed: 30 May 2023).

APPENDIX

I encountered the below error with installing MongoDB as I used Ubuntu Version 22.04.
It appears that the issue was with Ubuntu 22.04, so I had to use Ubuntu 20.04. Here, I used two approaches to this: I installed a new disc image for Ubuntu 20.04 in my Virtual Box and had this running in my terminal but a quicker and easier way to do this, is just to select the Application name and OS to Ubuntu 20.04 while setting up a new Instance in AWS.