Developing Course App - Backend

Working Steps :

A)Setup MongoDB

1) Create the Database called jlcdb

2) Create the Collection called mycourses

3) Insert 5 Documents into mycourses collection as follows

```
[{
"courseId": "C-101",
"courseName": "Java Full Stack Course",
"duration": "500 Hrs",
"trainer": "Srinivas Dande",
"enrollments": 25000
},
{
"courseId": "C-102",
"courseName": "Spring Boot Course",
"duration": "45 Hrs",
"trainer": "Srinivas Dande",
"enrollments": 200
},
{
"courseId": "C-103",
"courseName": "Spring MicroServices Course",
"duration": "45 Hrs",
"trainer": "Srinivas Dande",
"enrollments": 200
},
{
"courseId": "C-104",
"courseName": "Angular Course",
"duration": "45 Hrs",
"trainer": "Srinivas Dande",
"enrollments": 200
},
{
"courseId": "C-105",
"courseName": "React Course",
"duration": "45 Hrs",
"trainer": "Srinivas Dande",
"enrollments": 50
}]
```

B)Setup the backend Server with Node and Express

4) Create the folder for your application.

```
jlc-course-app-backend
```

5) Open that folder

```
jlc-course-app-backend>
```

6) Initialize the node project

```
jlc-course-app-backend> npm init
```

Walk through the questions asked and answer them

finally package.json will be created as follows.

```
{
  "name": "jlc-course-app-backend",
  "version": "1.1.1",
  "description": "This is JLC Course App Backend",
  "main": "myserver.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "ReactJS",
    "ExpressJS",
    "MongoDB"
  ],
  "author": "Srinivas Dande",
  "license": "ISC"
}
```

7) Install the Dependencies required for your backend project.

```
npm install express
npm install body-parser
npm install mongoose
npm install nodemon
npm install dotenv
npm install cors
```

8) Create .env.jlc file under the project folder

9) Place the following in .env.jlc file

```
BASE_URL=http://localhost:5500
MONGODB_URI=mongodb://localhost:27017/jlcdb
```

10) Create the file called myserver.js under the project folder

11) Place the following Code in myserver.js file

```
const express = require("express");
const dotenv = require("dotenv");

/**
 * Load environment variables from .env.jlc file.
 */
dotenv.config({ path: ".env.jlc" });

/**
 * Set Response for requestURI - /ping
 */
app.get("/ping", (req, res) => {
  console.log("Request for - /ping");
   return res.send("Hello Guys -- I am Ready");
});

const app = express();
const PORT = process.env.PORT || 5300;

/**
 * start Express server on port 5300
 */
app.listen(PORT, () => {
  console.log("Express server is running at http://localhost:%d", PORT);
  console.log("Press CTRL-C to stop\n");
});
```

12) Update the package.json

Before Updating:
```
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1"
 },
```

After Updating:
```
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "server": "nodemon --watch src myserver.js"
 },
```

13) Start the Server

```
npm run server
```

14) Open the URL

http://localhost:5300/hello
You can see Response ----- Hello Guys -- I am Ready

C) Develop the REST API

15) Create the src folder under the project folder

16) Create the models folder under src folder

17) Create the controllers folder under src folder

18) Write the Course Model under src/models

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

let Course = new Schema(
  {
    courseId: { type: String },
    courseName: { type: String },
    duration: { type: String },
    trainer: { type: String },
    enrollments: { type: Number },
  },
  { collection: "mycourses" }
);
module.exports = mongoose.model("Course", Course);
```

19) Write CourseController under controllers folder

```
const express = require("express");
const mycourseRoute = express.Router();

// Get All Courses
mycourseRoute.route("/mycourses").get(
  (req, res) => {
  console.log("/mycourses ---get()");
  Course.find((error, data) => {
    if (error) {
      return next(error);
    } else {
      res.json(data);
    }
  }).sort({ bookId: 1 });
 });

 // Get Course By Id
 mycourseRoute.route("/mycourse/:courseId").get((req, res, next) => {
  console.log("/mycourse/:courseId --- get()");
  Course.findOne({ courseId: req.params.courseId }, (error, data) => {
    if (error) {
      return next(error);
    } else {
      res.json(data);
    }
  });
 });

 module.exports = mycourseRoute;
```

20) Update myserver.js

```
const express = require("express");
const bodyParser = require("body-parser");
const dotenv = require("dotenv");
const mongoose = require("mongoose");
const cors = require("cors");

/**
 * Load environment variables from .env.jlc file.
 */
dotenv.config({ path: ".env.jlc" });
```

```
/**
 * Connect to MongoDB.
 */
mongoose
  .connect(process.env.MONGODB_URI, {
    useUnifiedTopology: true,
    useFindAndModify: false,
    useNewUrlParser: true,
  })
  .then(
    () => {
      console.log("MongoDB connected successfully.");
    },
    (error) => {
      console.error(error);
      console.log(
        "MongoDB connection error. Please make sure MongoDB is running."
      );
      process.exit();
    }
  );

/**
 * Add middlewares
 */
const app = express();
const PORT = process.env.PORT || 5300;

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cors());

const mycourseController = require("./src/controllers/CourseController");

app.use("/myapi/", mycourseController);

/**
 * Set Response for requestURI - /hello
 */
app.get("/hello", (req, res) => {
    console.log("Request for - /hello");
     return res.send("Hello Guys -- I am Ready");
  });
 /**
 * start Express server on port 5300
 */
app.listen(PORT, () => {
   console.log("Express server is running at http://localhost:%d", PORT);
   console.log("Press CTRL-C to stop\n");
 });
```

21) Start the Server

```
npm run server
```

22) Test API

http://localhost:5300/myapi/mycourses
http://localhost:5300/myapi/mycourse/C-101

Course App - Backend

Files Required

1. Course.js	2. CourseController.js
3. .env.jlc	4. myserver.js
5. package.json	

1. Course.js

```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;

// Define collection and schema
let Course = new Schema(
  {
    courseId: { type: String },
    courseName: { type: String },
    duration: { type: String },
    trainer: { type: String },
    enrollments: { type: Number },
  },
  { timestamps: true, collection: "mycourses" }
);
module.exports = mongoose.model("Course", Course);
```

2. CourseController.js

```
const express = require("express");
const mycourseRoute = express.Router();

let Course = require("../models/Course");

// Get All Courses
mycourseRoute.route("/mycourses").get(
  (req, res) => {
  console.log("/mycourses ---get()");
  Course.find((error, data) => {
   if (error) {
     return next(error);
   } else {
     res.json(data);
   }
  }).sort({ bookId: 1 });
 });

 // Get Course By Id
 mycourseRoute.route("/mycourse/:courseId").get((req, res, next) => {
  console.log("/mycourse/:courseId --- get()");
  Course.findOne({ courseId: req.params.courseId }, (error, data) => {
   if (error) {
     return next(error);
   } else {
     res.json(data);
   }
  });
 });

 module.exports = mycourseRoute;
```

3. .env.jlc

```
const express = require("express");
const mycourseRoute = express.Router();
```

4. myserver.js

```
const express = require("express");
const bodyParser = require("body-parser");
const dotenv = require("dotenv");
const mongoose = require("mongoose");
const cors = require("cors");
```

```
/**
 * Load environment variables from .env.jlc file.
 */
dotenv.config({ path: ".env.jlc" });

/**
 * Connect to MongoDB.
 */
mongoose
  .connect(process.env.MONGODB_URI, {
    useUnifiedTopology: true,
    useFindAndModify: false,
    useNewUrlParser: true,
  })
  .then(
    () => {
      console.log("MongoDB connected successfully.");
    },
    (error) => {
      console.error(error);
      console.log(
        "MongoDB connection error. Please make sure MongoDB is running."
      );
      process.exit();
    }
  );

/**
 * Add middlewares
 */
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cors());

const mycourseController = require("./src/controllers/CourseController");

app.use("/myapi/", mycourseController);

/**
 * Set Response for requestURI - /hello
 */
app.get("/hello", (req, res) => {
    console.log("Request for - /hello");
     return res.send("Hello Guys -- I am Ready");
  });
 /**
* start Express server on port 5300
*/

const PORT = process.env.PORT || 5300;
app.listen(PORT, () => {
  console.log("Hello, Express server is running at http://localhost:%d", PORT);
  console.log("Press CTRL-C to stop\n");
 });
```

5. package.json

```
{
 "name": "jlc-course-app-backend",
 "version": "1.1.1",
 "description": "This is JLC Course App Backend",
 "main": "myserver.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "server": "nodemon --watch src myserver.js"
 },
 "keywords": [
  "ReactJS",
  "ExpressJS",
  "MongoDB"
 ],
 "author": "Srinivas Dande",
 "license": "ISC",
 "dependencies": {
  "body-parser": "^1.19.0",
  "cors": "^2.8.5",
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "mongoose": "^5.10.6",
  "nodemon": "^2.0.4"
 }
}
```

Developing FrontEnd with React

Working Steps:

1. **Create the React Project**

```
create-react-app react-course-app-frontend --scripts-version 1.1.5
```

2. **Enable Boorstrap in React App.**

a)Install Jquery

```
npm install jquery bootstrap
```

b)import the bootstrap.min.css in App.js

```
import "bootstrap/dist/css/bootstrap.min.css";
```

3. **Install Packages required for Your Project.**

```
npm install axios --save
```

4. **Update Index.js**

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

5. **Write CourseList Component**
   CourseList.js

6. **Update App.js**

```
import React, { Component } from 'react';

import "bootstrap/dist/css/bootstrap.min.css";
import CourseList from './CourseList';

class App extends Component {
  render() {
    return (
      <div className="App">
        <h1 className="text-center"> JLC BookStore </h1>
        <br/>
```

```
            <CourseList/>
          </div>
        );
      }
    }

export default App;
```

7. Run the App

```
npm start
```

8. Open the URL

http://localhost:3000/

Course App - Frontend

Files Required

1. index.js	2. App.js
3. CourseList.js	4. package.json

1. index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

2. App.js

```
import React, { Component } from 'react';
import "bootstrap/dist/css/bootstrap.min.css";
import CourseList from './CourseList';

class App extends Component {
  render() {
    return (
      <div className="App">
        <h1 className="text-center"> JLC BookStore </h1>
        <br/>
        <CourseList/>
      </div>
    );
  }
}
export default App;
```

3. CourseList.js

```
import React, { Component } from "react";

import axios from 'axios';

class CourseList extends Component {

 state={
  mycourses:[]
 }
 componentDidMount(){
  console.log("componentDidMount");
  //const URL="https://coursecube.com/jlc-rest-api/mini-courses";
  const URL="http://localhost:5300/myapi/mycourses";
  axios.get(URL)
  .then(myresponse => {
   this.setState({mycourses:myresponse.data});
  });

 }
 render() {
  const mycourseList= this.state.mycourses.map(
   mycourse => {
    return (
     <tr key={mycourse.courseId}>
      <td> {mycourse.courseId} </td>
      <td> {mycourse.courseName} </td>
      <td> {mycourse.duration} </td>
      <td> {mycourse.trainer} </td>
      <td> {mycourse.enrollments} </td>
     </tr>
    );
   }
  );
  return (
   <div className="container">
    <br />
    <table className="table">
     <thead>
      <tr>
       <th> Course ID</th>
       <th> Course Name</th>
       <th> Duration</th>
       <th> Trainer</th>
       <th> Enrollments</th>
       <th> </th>
```

```
          </tr>
          </thead>
          <tbody>
           {mycourseList}
          </tbody>
        </table>
       </div>
      );
     }
    }
    export default CourseList;
```

4. package.json

```
{
  "name": "react-course-app-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "axios": "^0.20.0",
    "bootstrap": "^4.5.2",
    "jquery": "^3.5.1",
    "react": "^16.13.1",
    "react-dom": "^16.13.1",
    "react-scripts": "1.1.5"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

Developing MERN Stack Application

1. **Create the React Project**

```
create-react-app react-course-app-mern --scripts-version 1.1.5
```

2. **Enable Boorstrap in React App.**

   a)Install Jquery

```
npm install jquery bootstrap
```

   b)import the bootstrap.min.css in App.js

```
import "bootstrap/dist/css/bootstrap.min.css";
```

3. **Install Packages required for Your React Project.**

```
npm install axios --save
```

4. **Install the Dependencies required for your backend project.**

```
npm install express
npm install body-parser
npm install mongoose
npm install nodemon
npm install dotenv
npm install cors
```

5. **Copy the following into src folder**
   a) index.js
   b) CourseLIst.js
   c) App.js

6. **Copy the following 2 files under react-course-app-mern**
   a) .env.jlc
   b) myserver.js

7. **Ceate the folder called myserver under react-course-app-mern**

8. **copy the following two folders under myserver**
   a) myserver/models/Course.js
   b) myserver/controllers/CourseController.js

9. Update myserver.js

```
const path = require("path");
....
app.use(express.static(path.join(__dirname, "build")));
....
 app.get("*", (req, res) => {
   res.sendFile(
     path.join(__dirname, "build", "index.html")
   );
 });
```

10. Update package.json

```
{
 "name": "react-course-app-mern",
 "version": "0.1.0",
 "description": "This is the MEAN App",
 "main": "myserver.js",
 "private": true,
 "keywords": [
   "NodeJS",
   "ExpressJS",
   "MongoDB",
   "Angular"
 ],
 "author": "Srinivas Dande",
 "license": "ISC",
 "dependencies": {
         ....
 },
 "proxy": "http://localhost:5300",
   "scripts": {
          .....
   "eject": "react-scripts eject",
   "server": "nodemon --watch myserver.js"
 }
}
```

11. Build Project

```
npm run build
```

12. Run the App

```
npm run server
      or
 node myserver.js
```

13. Open http://localhost:5300/

Course App - MERN Stack

Files Required

1. index.js	Same as Frontend
2. App.js	Same as Frontend
3. BooksList.js	Same as Frontend
4. Course.js	Same as Frontend
5. CourseController.js	Same as Frontend
6. .env.jlc	Same as Frontend
7. myserver.js	Updated
8. package.json	Updated

7. myserver.js

```
const express = require("express");
const bodyParser = require("body-parser");
const dotenv = require("dotenv");
const mongoose = require("mongoose");
const cors = require("cors");
const path = require("path");

/**
 * Load environment variables from .env.jlc file.
 */
dotenv.config({ path: ".env.jlc" });

/**
 * Connect to MongoDB.
 */
mongoose
 .connect(process.env.MONGODB_URI, {
   useUnifiedTopology: true,
   useFindAndModify: false,
   useNewUrlParser: true,
 })
 .then(
   () => {
     console.log("MongoDB connected successfully.");
   },
   (error) => {
     console.error(error);
     console.log(
       "MongoDB connection error. Please make sure MongoDB is running."
     );
```

```
      process.exit();
    }
  };

/**
 * Add middlewares
 */
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static(path.join(__dirname, "build")));
app.use(cors());

const mycourseController = require("./myserver/controllers/CourseController");
app.use("/myapi/", mycourseController);

/**
 * Set Response for requestURI - /hello
 */
app.get("/hello", (req, res) => {
    console.log("Request for - /hello");
     return res.send("Hello Guys -- I am Ready");
  });

  app.get("*", (req, res) => {
    res.sendFile(
      path.join(__dirname, "build", "index.html")
    );
  });

  /**
 * start Express server on port 5300
 */

const PORT = process.env.PORT || 5300;
app.listen(PORT, () => {
    console.log("Hello, Express server is running at http://localhost:%d", PORT);
    console.log("Press CTRL-C to stop\n");
  });
```

8. package.json

```
{
 "name": "react-course-app-mern",
 "version": "0.1.0",
 "description": "This is the MEAN App",
 "main": "myserver.js",
 "private": true,
 "keywords": [
  "NodeJS",
  "ExpressJS",
  "MongoDB",
  "Angular"
 ],
 "author": "Srinivas Dande",
 "license": "ISC",
 "dependencies": {
  "axios": "^0.20.0",
  "body-parser": "^1.19.0",
  "bootstrap": "^4.5.2",
  "cors": "^2.8.5",
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "jquery": "^3.5.1",
  "mongoose": "^5.10.6",
  "nodemon": "^2.0.4",
  "react": "^16.13.1",
  "react-dom": "^16.13.1",
  "react-scripts": "1.1.5"
 },
 "proxy": "http://localhost:5300",
  "scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test --env=jsdom",
  "eject": "react-scripts eject",
  "server": "nodemon --watch myserver.js"
 }
}
```