Here’s a step-by-step guide to complete Assignment 4:

**Assignment 4a: Create a Node.js Application That Serves a Static Website**

**1. Set Up the Node.js Application**

* Initialize a new Node.js project:

bash

Copy code

mkdir node-static-site

cd node-static-site

npm init -y

* Install the necessary packages:

bash

Copy code

npm install express

**2. Create the Project Structure**

* Project structure:

java

Copy code

node-static-site/

├── public/

│ ├── index.html

│ ├── about.html

│ └── style.css

├── server.js

└── package.json

* Create a public folder with HTML files (e.g., index.html and about.html).

**3. Serve Static Files Using Express.js**

* server.js file:

javascript

Copy code

const express = require('express');

const app = express();

const port = 3000;

// Serve static files from the 'public' directory

app.use(express.static('public'));

app.listen(port, () => {

console.log(`Server is running at http://localhost:${port}`);

});

* Add a simple HTML page (public/index.html):

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Static Site</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Welcome to the Static Website</h1>

<p>This is the homepage.</p>

</body>

</html>

* Start the server:

bash

Copy code

node server.js

* Visit http://localhost:3000 to view your static website.

**Assignment 4b: Create CRUD APIs Using Node.js, Express.js, and MongoDB**

**1. Set Up the Node.js Application**

* Create a new directory for the project:

bash

Copy code

mkdir node-crud-api

cd node-crud-api

npm init -y

* Install the necessary dependencies:

bash

Copy code

npm install express mongoose body-parser cors

**2. Set Up MongoDB**

* If you haven’t installed MongoDB, you can set it up locally or use a cloud-based solution like [MongoDB Atlas](https://www.mongodb.com/cloud/atlas).

**3. Create the Project Structure**

* Project structure:

go

Copy code

node-crud-api/

├── models/

│ └── Assignment.js

├── routes/

│ └── assignmentRoutes.js

├── server.js

└── package.json

**4. Create a Mongoose Model**

* models/Assignment.js:

javascript

Copy code

const mongoose = require('mongoose');

const assignmentSchema = new mongoose.Schema({

title: { type: String, required: true },

description: String,

dueDate: Date,

status: { type: String, default: 'Pending' }

});

module.exports = mongoose.model('Assignment', assignmentSchema);

**5. Create Express Routes for CRUD Operations**

* routes/assignmentRoutes.js:

javascript

Copy code

const express = require('express');

const router = express.Router();

const Assignment = require('../models/Assignment');

// Create a new assignment (Create)

router.post('/assignments', async (req, res) => {

try {

const newAssignment = new Assignment(req.body);

await newAssignment.save();Here’s a step-by-step guide to complete Assignment 4:

**Assignment 4a: Create a Node.js Application That Serves a Static Website**

**1. Set Up the Node.js Application**

* Initialize a new Node.js project:

bash

Copy code

mkdir node-static-site

cd node-static-site

npm init -y

* Install the necessary packages:

bash

Copy code

npm install express

**2. Create the Project Structure**

* Project structure:

java

Copy code

node-static-site/

├── public/

│ ├── index.html

│ ├── about.html

│ └── style.css

├── server.js

└── package.json

* Create a public folder with HTML files (e.g., index.html and about.html).

**3. Serve Static Files Using Express.js**

* server.js file:

javascript

Copy code

const express = require('express');

const app = express();

const port = 3000;

// Serve static files from the 'public' directory

app.use(express.static('public'));

app.listen(port, () => {

console.log(`Server is running at http://localhost:${port}`);

});

* Add a simple HTML page (public/index.html):

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Static Site</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Welcome to the Static Website</h1>

<p>This is the homepage.</p>

</body>

</html>

* Start the server:

bash

Copy code

node server.js

* Visit http://localhost:3000 to view your static website.

**Assignment 4b: Create CRUD APIs Using Node.js, Express.js, and MongoDB**

**1. Set Up the Node.js Application**

* Create a new directory for the project:

bash

Copy code

mkdir node-crud-api

cd node-crud-api

npm init -y

* Install the necessary dependencies:

bash

Copy code

npm install express mongoose body-parser cors

**2. Set Up MongoDB**

* If you haven’t installed MongoDB, you can set it up locally or use a cloud-based solution like [MongoDB Atlas](https://www.mongodb.com/cloud/atlas).

**3. Create the Project Structure**

* Project structure:

go

Copy code

node-crud-api/

├── models/

│ └── Assignment.js

├── routes/

│ └── assignmentRoutes.js

├── server.js

└── package.json

**4. Create a Mongoose Model**

* models/Assignment.js:

javascript

Copy code

const mongoose = require('mongoose');

const assignmentSchema = new mongoose.Schema({

title: { type: String, required: true },

description: String,

dueDate: Date,

status: { type: String, default: 'Pending' }

});

module.exports = mongoose.model('Assignment', assignmentSchema);

**5. Create Express Routes for CRUD Operations**

* routes/assignmentRoutes.js:

javascript

Copy code

const express = require('express');

const router = express.Router();

const Assignment = require('../models/Assignment');

// Create a new assignment (Create)

router.post('/assignments', async (req, res) => {

try {

const newAssignment = new Assignment(req.body);

await newAssignment.save();

res.status(201).json(newAssignment);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

// Read all assignments (Read)

router.get('/assignments', async (req, res) => {

try {

const assignments = await Assignment.find();

res.json(assignments);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

// Update an assignment (Update)

router.put('/assignments/:id', async (req, res) => {

try {

const updatedAssignment = await Assignment.findByIdAndUpdate(req.params.id, req.body, { new: true });

if (!updatedAssignment) return res.status(404).json({ message: 'Assignment not found' });

res.json(updatedAssignment);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

// Delete an assignment (Delete)

router.delete('/assignments/:id', async (req, res) => {

try {

const deletedAssignment = await Assignment.findByIdAndDelete(req.params.id);

if (!deletedAssignment) return res.status(404).json({ message: 'Assignment not found' });

res.json({ message: 'Assignment deleted' });

} catch (error) {

res.status(500).json({ message: error.message });

}

});

module.exports = router;

**6. Set Up the Server**

* server.js:

javascript

Copy code

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const assignmentRoutes = require('./routes/assignmentRoutes');

const app = express();

const port = 3000;

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/assignmentsDB', {

useNewUrlParser: true,

useUnifiedTopology: true,

useFindAndModify: false

}).then(() => console.log('Connected to MongoDB'))

.catch(err => console.error(err));

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Routes

app.use('/api', assignmentRoutes);

// Start the server

app.listen(port, () => {

console.log(`Server is running at http://localhost:${port}`);

});

**7. Testing the API**

* Use tools like **Postman** or **cURL** to test the following routes:
  + POST /api/assignments - Create a new assignment.
  + GET /api/assignments - Retrieve all assignments.
  + PUT /api/assignments/:id - Update an assignment by ID.
  + DELETE /api/assignments/:id - Delete an assignment by ID.

**Summary**

* **Part A**: Created a simple Node.js server serving static files using Express.
* **Part B**: Built four CRUD APIs using Node.js, Express.js, and MongoDB to manage assignments data.

This structure should give you a clear pathway to completing your Node.js assignments. Happy coding! 🚀

res.status(201).json(newAssignment);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

// Read all assignments (Read)

router.get('/assignments', async (req, res) => {

try {

const assignments = await Assignment.find();

res.json(assignments);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

// Update an assignment (Update)

router.put('/assignments/:id', async (req, res) => {

try {

const updatedAssignment = await Assignment.findByIdAndUpdate(req.params.id, req.body, { new: true });

if (!updatedAssignment) return res.status(404).json({ message: 'Assignment not found' });

res.json(updatedAssignment);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

// Delete an assignment (Delete)

router.delete('/assignments/:id', async (req, res) => {

try {

const deletedAssignment = await Assignment.findByIdAndDelete(req.params.id);

if (!deletedAssignment) return res.status(404).json({ message: 'Assignment not found' });

res.json({ message: 'Assignment deleted' });

} catch (error) {

res.status(500).json({ message: error.message });

}

});

module.exports = router;

**6. Set Up the Server**

* server.js:

javascript

Copy code

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const assignmentRoutes = require('./routes/assignmentRoutes');

const app = express();

const port = 3000;

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/assignmentsDB', {

useNewUrlParser: true,

useUnifiedTopology: true,

useFindAndModify: false

}).then(() => console.log('Connected to MongoDB'))

.catch(err => console.error(err));

// Middleware

app.use(cors());

app.use(bodyParser.json());

// Routes

app.use('/api', assignmentRoutes);

// Start the server

app.listen(port, () => {

console.log(`Server is running at http://localhost:${port}`);

});

**7. Testing the API**

* Use tools like **Postman** or **cURL** to test the following routes:
  + POST /api/assignments - Create a new assignment.
  + GET /api/assignments - Retrieve all assignments.
  + PUT /api/assignments/:id - Update an assignment by ID.
  + DELETE /api/assignments/:id - Delete an assignment by ID.

**Summary**

* **Part A**: Created a simple Node.js server serving static files using Express.
* **Part B**: Built four CRUD APIs using Node.js, Express.js, and MongoDB to manage assignments data.

This structure should give you a clear pathway to completing your Node.js assignments. Happy coding! 🚀