Assignment-4

Assignment-4

Simple Note-Taking Application Using MERN

Objective: Create a basic note-taking application using the MERN stack that allows users to create, read, update, and delete notes.

Tasks:

Backend (Node.js + Express.js + MongoDB)

1. Setup and Initialization

1. Create a new Node.js project:

bash

Copy code

mkdir mern-note-taking

cd mern-note-taking

npm init -y

2. Install necessary packages:

bash

Copy code

npm install express mongoose body-parser cors dotenv

3. Ensure the application starts without errors:

Create app.js:

javascript

Copy code

// app.js

const express = require('express');

const mongoose = require('mongoose');

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

const cors = require('cors');

require('dotenv').config();

```
const app = express();
const port = process.env.PORT || 5000;

// Middleware
app.use(cors());
app.use(bodyParser.json());

// MongoDB connection
mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error('MongoDB connection error:', err));

// Routes
app.get('/', (req, res) => {
  res.send('Welcome to the Note-Taking Application');
});

// Start server
app.listen(port, () => {
  console.log(Server is running on http://localhost:${port});
});
```

4. Create a .env file:

env

Copy code

```
PORT=5000
MONGODB_URI=your_mongodb_connection_string
```

5. Run the server:

bash

Copy code

node app.js

2. Create the Express Server

The server setup is included in the app.js file above.

3. Configure MongoDB

Set up a MongoDB database using MongoDB Atlas or a local MongoDB server, and update the .env file with the connection string.

4. Define Note Model

1. Create a models directory and a Note.js file:

bash

Copy code

mkdir models

touch models/Note.js

2. Define the Note model:

javascript

Copy code

```
// models/Note.js
const mongoose = require('mongoose');

const NoteSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true
  },
  content: {
    type: String,
    required: true
  }
});
```

```
module.exports = mongoose.model('Note', NoteSchema);
```

5. Define Note Routes

1. Create a routes directory and a note.js file:

bash

Copy code

```
mkdir routes
touch routes/note.js
```

2. Define the routes:

javascript

Copy code

```
// routes/note.js
const express = require('express');
const router = express.Router();
const Note = require('../models/Note');

// GET /notes - Retrieve all notes
router.get('/notes', async (req, res) => {
  try {
    const notes = await Note.find();
    res.json(notes);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

// GET /notes/:id - Retrieve a note by ID
router.get('/notes/:id', async (req, res) => {
  try {
```

```
        const note = await Note.findById(req.params.id);
        if (!note) return res.status(404).json({ message: 'Note not found' });
        res.json(note);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});

// POST /notes - Create a new note
router.post('/notes', async (req, res) => {
    const note = new Note({
        title: req.body.title,
        content: req.body.content
    });
    try {
        const newNote = await note.save();
        res.status(201).json(newNote);
    } catch (err) {
        res.status(400).json({ message: err.message });
    }
});

// PUT /notes/:id - Update a note by ID
router.put('/notes/:id', async (req, res) => {
    try {
        const note = await Note.findById(req.params.id);
        if (!note) return res.status(404).json({ message: 'Note not found' });

        note.title = req.body.title;
```

```
    note.content = req.body.content;
    await note.save();
    res.json(note);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// DELETE /notes/:id - Delete a note by ID
router.delete('/notes/:id', async (req, res) => {
  try {
    const note = await Note.findById(req.params.id);
    if (!note) return res.status(404).json({ message: 'Note not found' });

    await note.remove();
    res.json({ message: 'Note deleted' });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

module.exports = router;
```

3. Integrate the routes with the Express server:

Update app.js:

javascript

Copy code

```
// app.js
// ... previous code
```

```
const noteRoutes = require('./routes/note');
app.use('/api', noteRoutes);
```

6. Basic Error Handling

Error handling has been implemented in the routes defined above.

Frontend (React.js)

1. Setup and Initialization

1. Create a new React application:

bash

Copy code

```
npx create-react-app note-taking-app
cd note-taking-app
npm start
```

2. Create Components

1. Create a components directory:

bash

Copy code

```
mkdir src/components
```

2. Create the following components:

- NoteList.js:

javascript

Copy code

```
// src/components/NoteList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import NoteItem from './NoteItem';
import AddNote from './AddNote';

const NoteList = () => {
  const [notes, setNotes] = useState([]);
```

```
  useEffect(() => {
    fetchNotes();
  }, []);

  const fetchNotes = async () => {
    try {
      const res = await axios.get('http://localhost:5000/api/notes');
      setNotes(res.data);
    } catch (err) {
      console.error(err);
    }
  };

  return (
    <div>
      <h1>Notes</h1>
      <AddNote fetchNotes={fetchNotes} />
      {notes.map(note => (
        <NoteItem key={note._id} note={note} fetchNotes={fetchNotes} />
      ))}
    </div>
  );
};

export default NoteList;
```

- NoteItem.js:

javascript

Copy code

```
// src/components/NoteItem.js
import React from 'react';
import axios from 'axios';

const NoteItem = ({ note, fetchNotes }) => {
  const deleteNote = async (id) => {
    try {
      await axios.delete(http://localhost:5000/api/notes/${id});
      fetchNotes();
    } catch (err) {
      console.error(err);
    }
  };

  return (
    <div>
      <h2>{note.title}</h2>
      <p>{note.content}</p>
      <button onClick={() => deleteNote(note._id)}>Delete</button>
    </div>
  );
};

export default NoteItem;
```

- AddNote.js:

```
javascript
Copy code
// src/components/AddNote.js
import React, { useState } from 'react';
```

```
import axios from 'axios';

const AddNote = ({ fetchNotes }) => {
  const [title, setTitle] = useState('');
  const [content, setContent] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post('http://localhost:5000/api/notes', { title, content });
      setTitle('');
      setContent('');
      fetchNotes();
    } catch (err) {
      console.error(err);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Title"
        value={title}
        onChange={(e) => setTitle(e.target.value)}
      />
      <textarea
        placeholder="Content"
        value={content}
```

```
            onChange={(e) => setContent(e.target.value)}
          ></textarea>
          <button type="submit">Add Note</button>
        </form>
    );
};

export default AddNote;
```

3. Set Up State Management

State management is already handled using useState and useEffect hooks in the NoteList component.

4. API Integration

API integration is handled using axios in the components.

5. Styling

1. Apply basic CSS styling:

css

Copy code

```
/* src/App.css */
body {
    font-family: Arial, sans-serif;
}

h1 {
    text-align: center;
}

form {
    display: flex;
    flex-direction: column;
    align-items: center;
```

```
  margin-bottom: 20px;
}

input, textarea {
  width: 80%;
  margin-bottom: 10px;
  padding: 10px;
  font-size: 16px;
}

button {
  padding: 10px 20px;
  font-size: 16px;
  cursor: pointer;
}

div {
  margin: 20px auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  width: 80%;
}
```

2. Update App.js to render NoteList component:

javascript

Copy code

```
// src/App.js
import React from 'react';
import './App.css';
```

```
import NoteList from './components/NoteList';

function App() {
   return (
      <div className="App">
         <NoteList />
      </div>
   );
}

export default App;
```