Title: Building a Simple User Authentication App with MERN Stack and Next.js

Overview

This document provides a comprehensive guide to building a simple user authentication app using the MERN (MongoDB, Express, React, Node.js) stack integrated with Next.js. The application will feature user registration, login, logout, protected routes, and authentication using MongoDB Atlas as the database.

Technology Stack

Frontend: Next.js, React, Material UI (MUI), React Router

Backend: Express.js, Node.js, MongoDB (Mongoose ORM)

Authentication: bcrypt (password hashing), JWT (JSON Web Token), Google OAuth (optional)

Database: MongoDB Atlas

State Management: React Context API (or Redux if needed)

Other Dependencies: Axios (for API calls), dotenv (for environment variables), cors (Cross-Origin Resource Sharing), express-validator (for input validation)

Project Structure

project-folder/

│── backend/

│ ├── config/

│ │ ├── db.js

│ ├── models/

│ │ ├── User.js

│ ├── routes/

│ │ ├── authRoutes.js

│ ├── middleware/

│ │ ├── authMiddleware.js

│ ├── controllers/

│ │ ├── authController.js

│ ├── server.js

│── frontend/

│ ├── components/

│ ├── pages/

│ │ ├── \_app.js

│ │ ├── index.js

│ │ ├── login.js

│ │ ├── register.js

│ │ ├── dashboard.js

│ ├── context/

│ │ ├── AuthContext.js

│ ├── styles/

│── .env

│── package.json

│── README.md

**Backend Setup (Express & MongoDB Atlas)**

1. **Initialize Backend**
2. mkdir backend && cd backend
3. npm init -y

npm install express mongoose dotenv bcryptjs jsonwebtoken cors express-validator

1. **Connect to MongoDB Atlas** (backend/config/db.js):
2. const mongoose = require('mongoose');
3. require('dotenv').config();
4. const connectDB = async () => {
5. try {
6. await mongoose.connect(process.env.MONGO\_URI, {
7. useNewUrlParser: true,
8. useUnifiedTopology: true,
9. });
10. console.log('MongoDB Connected');
11. } catch (err) {
12. console.error(err.message);
13. process.exit(1);
14. }
15. };

module.exports = connectDB;

1. **User Model** (backend/models/User.js):
2. const mongoose = require('mongoose');
3. const UserSchema = new mongoose.Schema({
4. name: String,
5. email: { type: String, unique: true, required: true },
6. password: { type: String, required: true },
7. });

module.exports = mongoose.model('User', UserSchema);

1. **Authentication Routes** (backend/routes/authRoutes.js):
2. const express = require('express');
3. const { registerUser, loginUser, getUserProfile } = require('../controllers/authController');
4. const { protect } = require('../middleware/authMiddleware');
5. const router = express.Router();
6. router.post('/register', registerUser);
7. router.post('/login', loginUser);
8. router.get('/profile', protect, getUserProfile);

module.exports = router;

1. **Start Express Server** (backend/server.js):
2. const express = require('express');
3. const dotenv = require('dotenv');
4. const cors = require('cors');
5. const connectDB = require('./config/db');
6. dotenv.config();
7. connectDB();
8. const app = express();
9. app.use(express.json());
10. app.use(cors());
11. app.use('/api/auth', require('./routes/authRoutes'));

app.listen(5000, () => console.log('Server running on port 5000'));

**Frontend Setup (Next.js & React)**

1. **Initialize Frontend**
2. npx create-next-app frontend
3. cd frontend

npm install axios react-router-dom context-api

1. **Auth Context** (frontend/context/AuthContext.js):
2. import { createContext, useState } from 'react';
3. const AuthContext = createContext();
4. export const AuthProvider = ({ children }) => {
5. const [user, setUser] = useState(null);
7. const login = async (email, password) => {
8. // Call API to authenticate
9. };
11. const logout = () => {
12. setUser(null);
13. };
15. return (
16. <AuthContext.Provider value={{ user, login, logout }}>
17. {children}
18. </AuthContext.Provider>
19. );
20. };

export default AuthContext;

1. **Login Page** (frontend/pages/login.js):
2. import { useState, useContext } from 'react';
3. import AuthContext from '../context/AuthContext';
4. export default function Login() {
5. const [email, setEmail] = useState('');
6. const [password, setPassword] = useState('');
7. const { login } = useContext(AuthContext);
9. const handleSubmit = async (e) => {
10. e.preventDefault();
11. await login(email, password);
12. };
14. return (
15. <form onSubmit={handleSubmit}>
16. <input type="email" value={email} onChange={(e) => setEmail(e.target.value)} />
17. <input type="password" value={password} onChange={(e) => setPassword(e.target.value)} />
18. <button type="submit">Login</button>
19. </form>
20. );

}

1. **Next.js API Route for Authentication** (frontend/pages/api/auth.js):
2. import axios from 'axios';
3. export default async function handler(req, res) {
4. if (req.method === 'POST') {
5. const { email, password } = req.body;
6. try {
7. const response = await axios.post('http://localhost:5000/api/auth/login', { email, password });
8. res.status(200).json(response.data);
9. } catch (error) {
10. res.status(400).json({ message: 'Authentication failed' });
11. }
12. }

}

**Deployment**

* **Backend:** Deploy on Render/Heroku
* **Frontend:** Deploy on Vercel
* **Database:** MongoDB Atlas

This document serves as a blueprint for building a MERN stack authentication system with Next.js and MongoDB Atlas.