```
Todo-App/
?
??? backend/
?   ??? controllers/
?   ?   ??? authController.js
?   ?   ??? todoController.js
?   ??? models/
?   ?   ??? User.js
?   ?   ??? Todo.js
?   ??? middleware/
?   ?   ??? authMiddleware.js
?   ??? routes/
?   ?   ??? authRoutes.js
?   ?   ??? todoRoutes.js
?   ??? .env
?   ??? server.js
?   ??? package.json
?   ??? config/
?       ??? db.js
?
??? frontend/
?   ??? public/
?   ?   ??? index.html
?   ??? src/
?       ??? components/
?       ?   ??? Auth/
?       ?   ?   ??? Login.js
?       ?   ?   ??? Signup.js
?       ?   ??? Todo/
?       ?       ??? TodoList.js
?       ?       ??? TodoForm.js
?       ??? context/
?       ?   ??? AuthContext.js
?       ??? pages/
?       ?   ??? TodoPage.js
?       ?   ??? HomePage.js
?       ??? App.js
?       ??? index.js
?       ??? package.json
?       ??? .env
??? README.md
```

Backend Code (Node.js, Express.js)

1. Install Dependencies

```
cd backend
npm install express mongoose bcryptjs jsonwebtoken dotenv cors
```

2. Database Configuration (config/db.js)

```
const mongoose = require('mongoose');
```

```
const connectDB = async () => {
    try {
        await mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, us
        console.log('MongoDB connected');
    } catch (error) {
        console.error(error.message);
        process.exit(1);
    }
};
module.exports = connectDB;
```

3. User Model (models/User.js)

```
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({
    firstName: { type: String, required: true },
    lastName: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
});

// Hash password before saving
userSchema.pre('save', async function (next) {
    if (!this.isModified('password')) return next();
    const salt = await bcrypt.genSalt(10);
    this.password = await bcrypt.hash(this.password, salt);
    next();
});

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

4. Todo Model (models/Todo.js)

```
const mongoose = require('mongoose');

const todoSchema = new mongoose.Schema({
    text: { type: String, required: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true
    createdAt: { type: Date, default: Date.now },
});

module.exports = mongoose.model('Todo', todoSchema);
```

5. Auth Controller (controllers/authController.js)

```
const User = require('../models/User');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');
```

```
exports.signup = async (req, res) => {
    const { firstName, lastName, email, password } = req.body;
    try {
        const user = new User({ firstName, lastName, email, password });
        await user.save();
        res.status(201).json({ message: 'User registered successfully' });
    } catch (error) {
        res.status(500).json({ error: 'Error registering user' });
    }
};

exports.login = async (req, res) => {
    const { email, password } = req.body;
    try {
        const user = await User.findOne({ email });
        if (!user) return res.status(400).json({ error: 'Invalid email or passwo

        const isMatch = await bcrypt.compare(password, user.password);
        if (!isMatch) return res.status(400).json({ error: 'Invalid email or pass

        const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, { ex
        res.json({ token });
    } catch (error) {
        res.status(500).json({ error: 'Error logging in' });
    }
};
```

6. Todo Controller (controllers/todoController.js)

```
const Todo = require('../models/Todo');

exports.getTodos = async (req, res) => {
    try {
        const todos = await Todo.find({ userId: req.user.id });
        res.json(todos);
    } catch (error) {
        res.status(500).json({ error: 'Error fetching todos' });
    }
};

exports.createTodo = async (req, res) => {
    try {
        const todo = new Todo({ text: req.body.text, userId: req.user.id });
        await todo.save();
        res.status(201).json(todo);
    } catch (error) {
        res.status(500).json({ error: 'Error creating todo' });
    }
};

exports.deleteTodo = async (req, res) => {
    try {
        await Todo.findByIdAndDelete(req.params.id);
        res.json({ message: 'Todo deleted' });
    } catch (error) {
```

```
        res.status(500).json({ error: 'Error deleting todo' });
    }
};
```

7. Auth Middleware (middleware/authMiddleware.js)

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
    const token = req.header('Authorization').replace('Bearer ', '');
    if (!token) return res.status(401).json({ error: 'No token provided' });

    try {
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        req.user = decoded;
        next();
    } catch (error) {
        res.status(401).json({ error: 'Invalid token' });
    }
};
```

8. Routes (routes/authRoutes.js)

```
const express = require('express');
const { signup, login } = require('../controllers/authController');
const router = express.Router();

router.post('/signup', signup);
router.post('/login', login);

module.exports = router;
```

9. Routes (routes/todoRoutes.js)

```
const express = require('express');
const { getTodos, createTodo, deleteTodo } = require('../controllers/todoControl
const auth = require('../middleware/authMiddleware');
const router = express.Router();

router.get('/', auth, getTodos);
router.post('/', auth, createTodo);
router.delete('/:id', auth, deleteTodo);

module.exports = router;
```

10. Server Setup (server.js)

```
const express = require('express');
const connectDB = require('./config/db');
const cors = require('cors');
```

```
require('dotenv').config();

const app = express();
connectDB();

app.use(cors());
app.use(express.json());
app.use('/api/auth', require('./routes/authRoutes'));
app.use('/api/todos', require('./routes/todoRoutes'));

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Here's how you can implement the **frontend** of your MERN Todo application.

Frontend Code (React.js)

1. Install Dependencies

cd frontend

```
npx create-react-app .
npm install axios react-router-dom
```

2. Configure React Router (src/App.js)

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes, Navigate } from 'react-router-do
import Login from './components/Auth/Login';
import Signup from './components/Auth/Signup';
import TodoPage from './pages/TodoPage';
import { AuthProvider, useAuth } from './context/AuthContext';

function App() {
    const { user } = useAuth();
```

```
    return (
        <AuthProvider>
            <Router>
                <Routes>
                    <Route path="/" element={!user ? <Login /> : <Navigate to="/
                    <Route path="/signup" element={!user ? <Signup /> : <Navigat
                    <Route path="/todo" element={user ? <TodoPage /> : <Navigate
                </Routes>
            </Router>
        </AuthProvider>
    );
}

export default App;
```

3. Authentication Context (src/context/AuthContext.js)

```
import React, { createContext, useState, useContext, useEffect } from 'react';
import axios from 'axios';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
    const [user, setUser] = useState(null);

    useEffect(() => {
        const token = localStorage.getItem('token');
        if (token) {
            setUser(token); // Placeholder logic, improve as needed
        }
    }, []);

    const login = async (email, password) => {
        const { data } = await axios.post('/api/auth/login', { email, password }
        localStorage.setItem('token', data.token);
        setUser(data.token);
    };

    const signup = async (userData) => {
        await axios.post('/api/auth/signup', userData);
    };

    const logout = () => {
        localStorage.removeItem('token');
        setUser(null);
    };

    return (
        <AuthContext.Provider value={{ user, login, signup, logout }}>
            {children}
        </AuthContext.Provider>
    );
};

export const useAuth = () => useContext(AuthContext);
```

4. Signup Component (src/components/Auth/Signup.js)

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

import { useAuth } from '../../context/AuthContext';
import { useNavigate } from 'react-router-dom';

const Signup = () => {
    const [form, setForm] = useState({ firstName: '', lastName: '', email: '', pa
    const { signup } = useAuth();
    const navigate = useNavigate();

    const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.valu

    const handleSubmit = async (e) => {
        e.preventDefault();
        await signup(form);
        navigate('/');
    };

    return (
        <form onSubmit={handleSubmit}>
            <input name="firstName" placeholder="First Name" onChange={handleChan
            <input name="lastName" placeholder="Last Name" onChange={handleChange
            <input type="email" name="email" placeholder="Email" onChange={handle
            <input type="password" name="password" placeholder="Password" onChang
            <button type="submit">Sign Up</button>
        </form>
    );
};

export default Signup;
```

5. Login Component (src/components/Auth/Login.js)

```
import React, { useState } from 'react';
import { useAuth } from '../../context/AuthContext';
import { useNavigate } from 'react-router-dom';

const Login = () => {
    const [form, setForm] = useState({ email: '', password: '' });
    const { login } = useAuth();
    const navigate = useNavigate();

    const handleChange = (e) => setForm({ ...form, [e.target.name]: e.target.valu

    const handleSubmit = async (e) => {
        e.preventDefault();
        await login(form.email, form.password);
        navigate('/todo');
    };
```

```
    return (
        <form onSubmit={handleSubmit}>
            <input type="email" name="email" placeholder="Email" onChange={handl
            <input type="password" name="password" placeholder="Password" onChan
            <button type="submit">Login</button>
        </form>
    );
};

export default Login;
```

6. Todo Page (src/pages/TodoPage.js)

import React, { useState, useEffect } from 'react';

```
import axios from 'axios';
import { useAuth } from '../context/AuthContext';

const TodoPage = () => {
    const { logout } = useAuth();
    const [todos, setTodos] = useState([]);
    const [text, setText] = useState('');

    const fetchTodos = async () => {
        const { data } = await axios.get('/api/todos', {
            headers: { Authorization: `Bearer ${localStorage.getItem('token')}` }
        });
        setTodos(data);
    };

    const addTodo = async (e) => {
        e.preventDefault();
        const { data } = await axios.post('/api/todos', { text }, {
            headers: { Authorization: `Bearer ${localStorage.getItem('token')}` }
        });
        setTodos([...todos, data]);
        setText('');
    };

    const deleteTodo = async (id) => {
        await axios.delete(`/api/todos/${id}`, {
            headers: { Authorization: `Bearer ${localStorage.getItem('token')}` }
        });
        setTodos(todos.filter((todo) => todo._id !== id));
    };

    useEffect(() => {
        fetchTodos();
    }, []);

    return (
        <div>
            <button onClick={logout}>Logout</button>
```

```
            <form onSubmit={addTodo}>
                <input value={text} onChange={(e) => setText(e.target.value)} pla
                <button type="submit">Add Todo</button>
            </form>
            <ul>
                {todos.map((todo) => (
                    <li key={todo._id}>
                        {todo.text} <button onClick={() => deleteTodo(todo._id)}>
                    </li>
                ))}
            </ul>
        </div>
    );
};

export default TodoPage;
```

7. Run Frontend and Backend Servers

- **Backend:**

cd backend

```
npm run start
```

- **Frontend:**

cd frontend

```
npm start
```