MERN stack authentication project with **Register, Login, Logout**, and **Dashboard**, including:

- Protected routes
- JWT-based authentication
- Navbar changes after login
- Full **React (Vite)** frontend and **Node.js + Express + MongoDB** backend
- Clear **folder structure**

```
mern-auth/
│
├── backend/
│   ├── config/
│   │   └── db.js
│   ├── controllers/
│   │   └── authController.js
│   ├── middleware/
│   │   └── authMiddleware.js
│   ├── models/
│   │   └── User.js
│   ├── routes/
│   │   └── authRoutes.js
│   ├── .env
│   ├── server.js
│   └── package.json
│
├── frontend/
│   ├── src/
│   │   ├── components/
│   │   │   ├── Navbar.jsx
│   │   ├── pages/
│   │   │   ├── Login.jsx
│   │   │   ├── Register.jsx
│   │   │   ├── Dashboard.jsx
│   │   ├── App.jsx
│   │   ├── main.jsx
│   │   └── utils/
│   │      └── PrivateRoute.jsx
│   ├── vite.config.js
│   ├── package.json
│
└── README.md
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const dotenv = require('dotenv');
const cors = require('cors');
const authRoutes = require('./routes/authRoutes');

dotenv.config();

const app = express();
app.use(cors());
app.use(express.json());
app.use('/api/auth', authRoutes);

mongoose.connect(process.env.MONGO_URI)
  .then(() => app.listen(5000, () => console.log('Server running on 5000')))
  .catch(err => console.log(err));
```

.env

```
MONGO_URI=mongodb://localhost:27017/mernauth
JWT_SECRET=yourSecretKey
```

User.js

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

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

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

authController.js

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

exports.register = async (req, res) => {
  const { name, email, password } = req.body;
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const user = await User.create({ name, email, password: hashedPassword });
    res.status(201).json({ message: 'User registered successfully' });
  } catch (err) {
    res.status(400).json({ error: 'User already exists' });
  }
};

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 credentials' });

    const match = await bcrypt.compare(password, user.password);
    if (!match) return res.status(400).json({ error: 'Invalid credentials' });

    const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, { expiresIn: '1d' });
    res.json({ token });
  } catch (err) {
    res.status(500).json({ error: 'Login failed' });
  }
```

```
exports.dashboard = (req, res) => {
  res.json({ message: 'Welcome to the dashboard' });
};
```

authroutes.js

```
const express = require('express');
const router = express.Router();
const authCtrl = require('../controllers/authController');
const authMiddleware = require('../middleware/authMiddleware');

router.post('/register', authCtrl.register);
router.post('/login', authCtrl.login);
router.get('/dashboard', authMiddleware, authCtrl.dashboard);

module.exports = router;
```

authmiddleware.js

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

module.exports = (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader) return res.status(401).json({ error: 'No token
provided' });

  const token = authHeader.split(' ')[1];
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded;
    next();
  } catch {
    res.status(401).json({ error: 'Invalid token' });
  }
};
```

App.jsx

```
import { Routes, Route } from 'react-router-dom';
import Register from './pages/Register';
import Login from './pages/Login';
import Dashboard from './pages/Dashboard';
import Navbar from './components/Navbar';
import PrivateRoute from './utils/PrivateRoute';

function App() {
  return (
    <>
      <Navbar />
      <Routes>
        <Route path="/" element={<Register />} />
        <Route path="/login" element={<Login />} />
        <Route element={<PrivateRoute />}>
          <Route path="/dashboard" element={<Dashboard />} />
        </Route>
      </Routes>
    </>
  );
}

export default App;
```

Navbar.jsx

```
import { Link, useNavigate } from 'react-router-dom';

function Navbar() {
  const navigate = useNavigate();
  const token = localStorage.getItem('token');

  const handleLogout = () => {
    localStorage.removeItem('token');
    navigate('/login');
  };

  return (
    <nav>
      <Link to="/">Register</Link> | <Link to="/login">Login</Link> |{'
'}
      {token && (
        <>
          <Link to="/dashboard">Dashboard</Link> |{' '}
          <button onClick={handleLogout}>Logout</button>
        </>
      )}
    </nav>
  );
}

export default Navbar;
```

Register.jsx

```
import { useState } from 'react';
import axios from 'axios';

function Register() {
  const [form, setForm] = useState({ name: '', email: '', password: ''
});

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

  const handleSubmit = async e => {
    e.preventDefault();
    await axios.post('http://localhost:5000/api/auth/register', form);
    alert('Registration successful');
  };

  return (
    <form onSubmit={handleSubmit}>
      <input name="name" placeholder="Name"
onChange={handleChange} />
      <input name="email" placeholder="Email"
onChange={handleChange} />
      <input name="password" type="password"
placeholder="Password" onChange={handleChange} />
      <button>Register</button>
    </form>
  );
}

export default Register;
```

Login.jsx

```
import { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';

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

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

  const handleSubmit = async e => {
    e.preventDefault();
    const res = await axios.post('http://localhost:5000/api/auth/login',
form);
    localStorage.setItem('token', res.data.token);
    navigate('/dashboard');
  };

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

export default Login;
```

Dashboard.jsx

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

function Dashboard() {
  const [msg, setMsg] = useState('');

  useEffect(() => {
    axios
      .get('http://localhost:5000/api/auth/dashboard', {
        headers: { Authorization: `Bearer
${localStorage.getItem('token')}` }
      })
      .then(res => setMsg(res.data.message))
      .catch(() => setMsg('Unauthorized'));
  }, []);

  return <h1>{msg}</h1>;
}

export default Dashboard;
```

Privateroute.jsx

```
import { Navigate, Outlet } from 'react-router-dom';

const PrivateRoute = () => {
  const token = localStorage.getItem('token');
  return token ? <Outlet /> : <Navigate to="/login" />;
};

export default PrivateRoute;
```