



COLLEGE CODE : 9602

COLLEGE NAME : Arunachala college of engineering for women DEPARTMENT : Artificial Intelligence and Data Science STUDENT NM-ID : bc197903cdad2beaff84ef4b90e243b7 ROLLNO : 960223243038

DATE : 6/10/2025

Completed the project named as Phase 5 Technology Project

NAME : FE-Login Authentication System

SUBMITTED BY

NAME : Durga R

MOBILE NO :6383513428

**Final Demo Walkthrough**

**Backend (Node.js + Express + MongoDB + JWT) backend/package.json**

{

"name": "fe-auth-backend",

"version": "1.0.0",

"main": "server.js", "scripts": {

"start": "node server.js", "dev": "nodemon server.js"

},

"dependencies": { "bcryptjs": "^2.4.3",

"cors": "^2.8.5",

"dotenv": "^16.0.0",

"express": "^4.18.2",

"jsonwebtoken": "^9.0.0",

"mongoose": "^7.0.0"

},

"devDependencies": { "nodemon": "^2.0.22"

}

}

### backend/.env.example

PORT=5000

MONGO\_URI=mongodb://localhost:27017/fe\_auth\_db JWT\_SECRET=your\_jwt\_secret\_here JWT\_EXPIRES\_IN=7d

### backend/config/db.js

const mongoose = require('mongoose'); const connectDB = async (mongoUri) => { try {

await mongoose.connect(mongoUri, { useNewUrlParser: true, useUnifiedTopology: true }); console.log('MongoDB connected');

} catch (err) {

console.error('MongoDB connection error:', err.message); process.exit(1);

}

};

module.exports = connectDB;

## backend/models/User.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({ name: { type: String, required: true },

email: { type: String, required: true, unique: true }, password: { type: String, required: true }, createdAt: { type: Date, default: Date.now }

});

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

## backend/middleware/auth.js

const jwt = require('jsonwebtoken'); module.exports = (req, res, next) => {

const token = req.header('Authorization')?.split(' ')[1];

if (!token) return res.status(401).json({ message: 'No token, authorization denied' }); try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET); req.user = decoded.user;

next();

} catch (err) {

return res.status(401).json({ message: 'Token is not valid' });

}

};

#### backend/routes/auth.js

const express = require('express');

const router = express.Router(); const bcrypt = require('bcryptjs'); const jwt = require('jsonwebtoken');

const User = require('../models/User'); const auth = require('../middleware/auth');

// POST /api/auth/signup router.post('/signup', async (req, res) => {

const { name, email, password } = req.body; try {

let user = await User.findOne({ email });

if (user) return res.status(400).json({ message: 'User already exists' });

user = new User({ name, email, password }); const salt = await bcrypt.genSalt(10); user.password = await bcrypt.hash(password, salt); await user.save();

const payload = { user: { id: user.id } };

const token = jwt.sign(payload, process.env.JWT\_SECRET, { expiresIn: process.env.JWT\_EXPIRES\_IN || '7d' });

res.json({ token });

} catch (err) { console.error(err.message); res.status(500).send('Server error');

}

});

// POST /api/auth/login router.post('/login', async (req, res) => { const { email, password } = req.body; try {

const user = await User.findOne({ email });

if (!user) return res.status(400).json({ message: 'Invalid credentials' });

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) return res.status(400).json({ message: 'Invalid credentials' });

const payload = { user: { id: user.id } };

const token = jwt.sign(payload, process.env.JWT\_SECRET, { expiresIn: process.env.JWT\_EXPIRES\_IN || '7d' });

res.json({ token });

} catch (err) { console.error(err.message); res.status(500).send('Server error');

}

});

// GET /api/auth/me

router.get('/me', auth, async (req, res) => { try {

const user = await User.findById(req.user.id).select('-password'); res.json(user);

} catch (err) { console.error(err.message); res.status(500).send('Server error');

}

});

module.exports = router;

## backend/server.js

require('dotenv').config();

const express = require('express'); const cors = require('cors');

const connectDB = require('./config/db'); const authRoutes = require('./routes/auth');

const app = express();

const PORT = process.env.PORT || 5000;

// Connect DB connectDB(process.env.MONGO\_URI);

app.use(cors()); app.use(express.json());

app.use('/api/auth', authRoutes);

app.get('/', (req, res) => res.send('Auth API running')); app.listen(PORT, () => console.log(`Server started on port ${PORT}`));

## backend/README.md

Instructions to run backend: cd backend

cp .env.example .env # fill .env

npm install npm run dev

**Frontend (React)**

#### frontend/package.json

{

"name": "fe-auth-frontend",

"version": "1.0.0", "private": true, "dependencies": { "axios": "^1.4.0",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-router-dom": "^6.14.1",

"react-scripts": "5.0.1"

},

"scripts": {

"start": "react-scripts start", "build": "react-scripts build"

}

}

## frontend/README.md

Run:

cd frontend npm install npm start

Make sure the backend is running at http://localhost:5000 or update apiBase in src/services/api.js.

## frontend/src/services/api.js

import axios from 'axios';

const api = axios.create({ baseURL: process.env.REACT\_APP\_API\_BASE || 'http://localhost:5000' });

// attach token automatically if present api.interceptors.request.use((config) => { const token = localStorage.getItem('token');

if (token) config.headers.Authorization = `Bearer ${token}`; return config;

});

export default api;

## frontend/src/context/AuthContext.js

import React, { createContext, useState, useEffect } from 'react'; import api from '../services/api';

export const AuthContext = createContext();

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

const [loading, setLoading] = useState(true);

const loadUser = async () => {

const token = localStorage.getItem('token'); if (!token) { setLoading(false); return; }

try {

const res = await api.get('/api/auth/me'); setUser(res.data);

} catch (err) {

console.error(err); localStorage.removeItem('token'); setUser(null);

} finally { setLoading(false);

}

};

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

const login = (token) => { localStorage.setItem('token', token); return loadUser();

};

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

};

return (

<AuthContext.Provider value={{ user, loading, login, logout }}>

{children}

</AuthContext.Provider>

);

};

## frontend/src/components/ProtectedRoute.js

import React, { useContext } from 'react'; import { Navigate } from 'react-router-dom';

import { AuthContext } from '../context/AuthContext';

export default function ProtectedRoute({ children }) { const { user, loading } = useContext(AuthContext); if (loading) return <div>Loading...</div>;

return user ? children : <Navigate to="/login" replace />;

}

# frontend/src/pages/Signup.js

import React, { useState, useContext } from 'react'; import api from '../services/api';

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

export default function Signup() {

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

const [error, setError] = useState(null); const { login } = useContext(AuthContext); const navigate = useNavigate();

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

e.preventDefault();

setError(null); try {

const res = await api.post('/api/auth/signup', form); await login(res.data.token);

navigate('/dashboard');

} catch (err) {

setError(err.response?.data?.message || 'Signup failed');

}

};

return (

<div style={{ maxWidth: 400, margin: '2rem auto' }}>

<h2>Signup</h2>

<form onSubmit={handleSubmit}>

<div>

<input name="name" placeholder="Name" value={form.name} onChange={handleChange} required />

</div>

<div>

<input name="email" placeholder="Email" value={form.email} onChange={handleChange} required />

</div>

<div>

<input name="password" type="password" placeholder="Password" value={form.password} onChange={handleChange} required />

</div>

<button type="submit">Create account</button>

{error && <p style={{ color: 'red' }}>{error}</p>}

</form>

</div>

);

}

## frontend/src/pages/Login.js

import React, { useState, useContext } from 'react'; import api from '../services/api';

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

export default function Login() {

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

const [error, setError] = useState(null); const { login } = useContext(AuthContext); const navigate = useNavigate();

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

e.preventDefault();

setError(null); try {

const res = await api.post('/api/auth/login', form); await login(res.data.token); navigate('/dashboard');

} catch (err) {

setError(err.response?.data?.message || 'Login failed');

}

};

return (

<div style={{ maxWidth: 400, margin: '2rem auto' }}>

<h2>Login</h2>

<form onSubmit={handleSubmit}>

<div>

<input name="email" placeholder="Email" value={form.email} onChange={handleChange} required />

</div>

<div>

<input name="password" type="password" placeholder="Password" value={form.password} onChange={handleChange} required />

</div>

<button type="submit">Login</button>

{error && <p style={{ color: 'red' }}>{error}</p>}

</form>

</div>

);

}

## frontend/src/pages/Dashboard.js

import React, { useContext } from 'react';

import { AuthContext } from '../context/AuthContext';

export default function Dashboard() {

const { user, logout } = useContext(AuthContext); return (

<div style={{ maxWidth: 700, margin: '2rem auto' }}>

<h2>Welcome, {user?.name}</h2>

<p>Email: {user?.email}</p>

<button onClick={logout}>Logout</button>

</div>

);

}

## frontend/src/App.js

import React from 'react';

import { BrowserRouter, Routes, Route, Link } from 'react-router-dom'; import { AuthProvider } from './context/AuthContext';

import Signup from './pages/Signup'; import Login from './pages/Login';

import Dashboard from './pages/Dashboard';

import ProtectedRoute from './components/ProtectedRoute';

function App() { return (

<AuthProvider>

<BrowserRouter>

<nav style={{ padding: 10 }}>

<Link to="/signup">Signup</Link> | <Link to="/login">Login</Link> | <Link to="/dashboard">Dashboard</Link>

</nav>

<Routes>

<Route path="/signup" element={<Signup />} />

<Route path="/login" element={<Login />} />

<Route path="/dashboard" element={<ProtectedRoute><Dashboard /></ProtectedRoute>} />

<Route path="/" element={<Login />} />

</Routes>

</BrowserRouter>

</AuthProvider>

);

}

export default App;

## frontend/src/index.js

import React from 'react';

import { createRoot } from 'react-dom/client'; import App from './App';

createRoot(document.getElementById('root')).render(<App />);

## README

Provide this short README in your repo root describing both apps and steps to run. # FE Login Authentication System

This repository contains a React frontend and a Node.js (Express + MongoDB) backend implementing email/password authentication using JWT.

## Run backend cd backend

cp .env.example .env # edit .env

npm install npm run dev

## Run frontend cd frontend npm install

npm start

**2) Project Report**

# Project Overview

This project is a **Full-Stack Login Authentication System** built using **React (frontend)** and **Node.js with Express (backend)**.

It allows users to **register**, **log in**, and **access secured pages** with authentication handled using **JWT (JSON Web Tokens)**.

The system ensures secure communication between client and server using **MongoDB Atlas** for database storage.

## Objectives

1. To implement a secure and user-friendly authentication system.
2. To demonstrate frontend-backend integration using REST APIs.
3. To store user credentials safely in a cloud database.

## Technologies Used

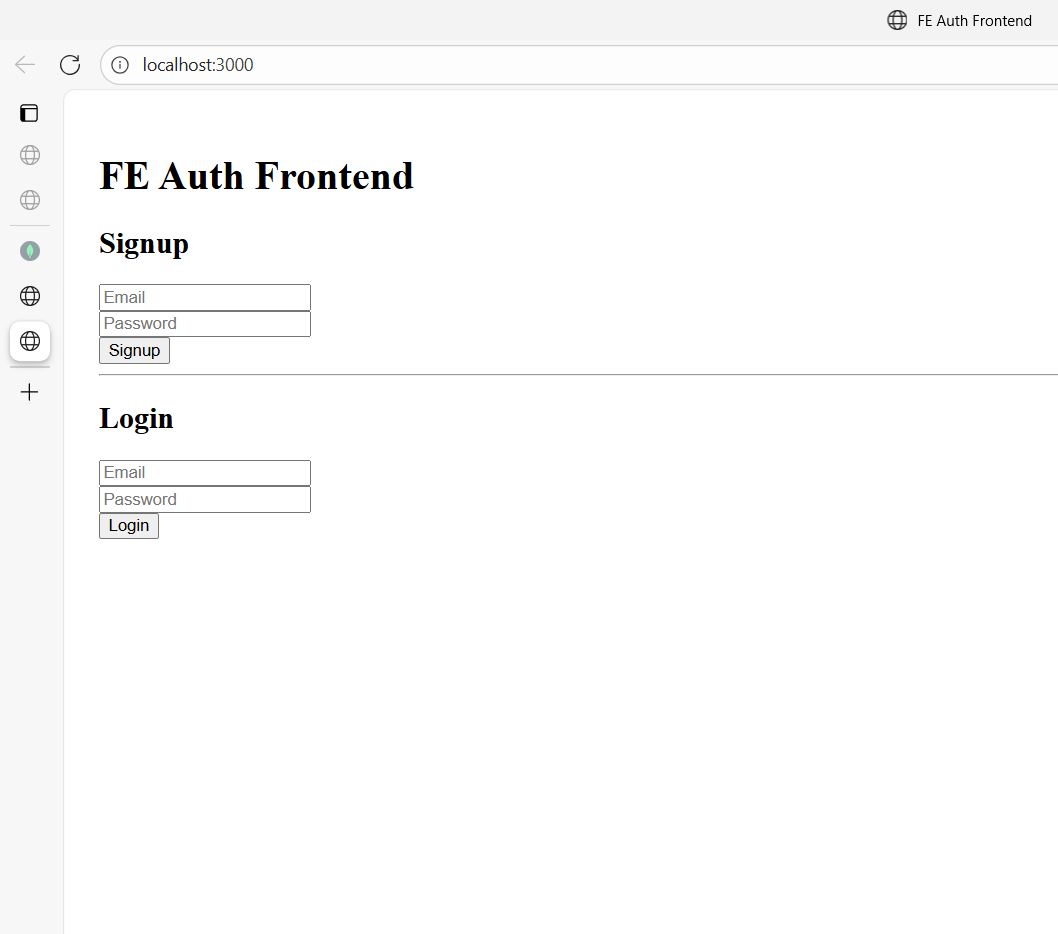
**Frontend:**React.js,Axios,CSS **Backend:**Node.js,Express.js **Database:**MongoDBAtlas

**Authentication:**JWT(JSONWebToken) **Version Control:** Git & GitHub

## Features

* 1. User Registration (Sign Up)
  2. Secure Login with JWT
  3. Password encryption using bcrypt
  4. Protected routes for authorized users
  5. Frontend-Backend communication via REST API

**Screenshot:**



**4.Challenges & Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| MongoDB connection issues | Fixed by updating the correct MongoDB connection string |
| Port conflicts (5000 already in use) | Solved by terminating running processes or using a different port |
| Missing dependencies during setup | Resolved using npm install and clearing  node\_modules |

**6.GitHub README**

http://localhost:3000/

**7. Final Submission (Repo + Deployed Link)**

https://github.com/durgaabi2803/Project.git