UserInfo Portal – Interview Q&A (DevOps Project)

Date: March 25, 2025

Candidate: Vamshi Panjala

Project: UserInfo Portal (DevOps + Full-Stack)

This document includes commonly asked interview questions based on the work completed in the UserInfo Portal project.

# Interview Questions & Answers – Day 1 & Day 2

* Q1: How did you structure your full-stack project?

A: We followed a company-style project layout with separated folders:  
- frontend/ for HTML/CSS/JS login UI  
- backend/ for Node.js + Express API  
- database/ for users.json (acts as DB)  
- scripts/, .github/, and backup/ for DevOps, automation, and tracking.

* Q2: How does your /login API work?

A: The API receives a POST request with email and password.  
It reads users.json, checks for matching credentials, and returns user details excluding the password.  
If no match, it returns an 'Invalid credentials' error.

* Q3: How did you test your backend API?

A: I used the REST Client extension in VS Code.  
I created a .http file to send POST requests to the /login endpoint with JSON input for email and password.

* Q4: How do you manage project versions and backups?

A: I used Git for version control.  
Initialized Git in the root folder, created a .gitignore file, and pushed everything to GitHub.  
Also maintained a local backup folder for daily snapshots and documentation.

* Q5: How would you explain this project in an interview?

A: This is a full-stack login system using Node.js, Express, and JSON as a temporary database.  
I built the backend API, tested it with REST Client, and used Git + GitHub for code backup.  
The structure follows real-world DevOps and team-based practices.

**Summary Update (Project Start → Now)**

Here’s what you can use as **team update or personal note**:

✅ **Project Name:** UserInfo Portal  
🎯 **Objective:** Build a working full-stack login system where users can securely log in and view their profile.

🚀 **What We’ve Done So Far:**

* Setup local folder structure (frontend, backend, database)
* Created login form in index.html
* Built backend API using Node.js + Express.js
* Used users.json for storing test users
* Connected frontend with backend using fetch() (AJAX)
* Fixed CORS errors with middleware in backend
* Stored session data in localStorage
* Displayed user info dynamically on dashboard.html
* Created GitHub repo and backed up daily progress
* Maintained clear documentation and Word reports

COMING PENDING **(Day 4):**

* Add Logout functionality
* Secure dashboard with auth redirection
* Finalize frontend session handling