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.

# Day 4 – Tasks Completed

1. Added Logout button to dashboard.html

2. Implemented logoutUser() function to clear localStorage and redirect to login

3. Protected dashboard from direct unauthorized access

4. Cleaned up duplicate JavaScript blocks

5. Full flow tested: Login → Dashboard → Logout

6. Final push to GitHub completed with clear commit message

# 🎯 Interview Questions & Answers (Day 4)

* Q1: What happens when a user clicks Logout?

A: The logoutUser() function clears the user session stored in localStorage and redirects the user to index.html (login page).

* Q2: How is the dashboard protected from direct access?

A: On page load, the dashboard checks if localStorage has a valid user object. If not found, it shows a message and redirects to login after 2 seconds.

* Q3: Why is localStorage used here?

A: It provides a simple, browser-based way to temporarily store session data for authenticated users.

* Q4: What improvements were made in code cleanup?

A: Removed duplicate script blocks and made the logout and user load logic more readable and modular.

* Q5: Why push to GitHub after each major update?

A: To ensure daily progress is backed up and can be tracked or reverted easily. Also good practice for team collaboration.

**Day-5 Questions:**

**Q1. How did you handle frontend form validation?**  
A: I validated inputs using JavaScript, checking email format, non-empty fields, and minimum password length.

**Q2. How did you show invalid credentials to users?**  
A: Instead of alerts, I showed an inline red error message below the login button that disappears after 3 seconds.

**Q3. How did you enhance user experience (UX)?**  
A: Autofocus on the email input and visual spacing helped improve usability.

**💡 Day-6 Questions:**

**Q4. How do you prevent dashboard access after logout?**  
A: I used window.history.forward() to block the browser back button from accessing cached pages.

**Q5. Where is session stored after login?**  
A: In localStorage as a JSON string containing user details.

**Q6. How is the welcome message shown dynamically?**  
A: The username is extracted from localStorage and inserted into the DOM using textContent.

**Q7. Did you test after every change?**  
A: Yes, login → dashboard → logout → back-button behavior was tested before pushing.

**Q8. Did you push to GitHub?**  
A: Yes, all changes are committed with proper messages and pushed to the main branch daily.

**Day-7 Interview Questions:**

**Project Overview:**

1. **Tell us about your project. What is it about, and what technologies did you use?**
   * Answer: "This project is a user login system that interacts with a MySQL database. The frontend uses HTML, CSS, and JavaScript, and the backend is built with Node.js using Express. The system allows users to log in with their email and password."

**Database Design:**

1. **How did you design the database schema?**
   * Answer: "I designed a simple table with necessary columns like name, email, password, phone, accountType, etc., to store user details. I used MySQL as the database because it's a reliable relational database for handling user information."

**SQL Queries:**

1. **Can you explain the SQL query used to insert data?**
   * Answer: "I used an INSERT INTO statement to add user data into the table. Here's the query:

sql

Copy

INSERT INTO users (name, email, password, phone, accountType, address, memberSince, lastLogin, status)

VALUES ('Vamshi Panjala', 'vamshi@gmail.com', 'Vamshi@123', '+91-9876543210', 'Premium', 'Hyderabad, Telangana, India', '2023-07-15', '2025-03-24 10:45 AM', 'Active');

This inserts a new user into the users table with the specified details."

**Backend Logic:**

1. **How does the login system work on the backend?**
   * Answer: "The backend uses a POST request to check the credentials against the database. If the email and password match, the backend sends a success response along with the user's details (excluding the password). If there's no match, an error message is returned."

**Frontend Interaction:**

1. **How does the frontend interact with the backend?**
   * Answer: "The frontend sends a POST request to the backend with the login credentials. The backend verifies the credentials and sends a response. If the login is successful, the frontend displays the user information; otherwise, it shows an error message."

**Security Considerations:**

1. **What security measures would you add to this system?**
   * Answer: "For security, I would hash the passwords before storing them in the database using bcrypt. Additionally, I would implement HTTPS for secure communication, JWT for session management, and rate-limiting to prevent brute-force attacks."

**Testing and Debugging:**

1. **How did you test the login system?**
   * Answer: "I tested the system by sending POST requests using Postman and checking the response. I also tested incorrect credentials to ensure that errors were properly handled."

**Day-8**

1. **How does your frontend handle invalid login inputs?**  
   → By validating email/password fields and showing message using a <div> with color styling.
2. **What happens when login credentials are wrong?**  
   → Backend returns a fail status, which is shown to the user as “Invalid credentials.”
3. **Why is frontend error handling important?**  
   → To give users immediate feedback, improve UX, and avoid sending bad data to the backend.
4. **What tools did you use for backup today?**  
   → Git (Git Bash), GitHub, and local backup/ folder with date-wise structure.
5. **Why did you skip bcrypt in this version?**  
   → Because this is a basic login system for practice only, not a production system.