# 📊 Live Demo Script – UserInfo Portal (Local Version)

📅 🎯 Introduction

Today I’m presenting the UserInfo Portal – a basic login system built using HTML, CSS, JavaScript (Frontend), Node.js + Express.js (Backend), and MySQL (Database). This demo will walk through each part of the project, explaining how everything works locally before deployment.

## 🗂️ Project Folder Structure

- Show main project folder with: frontend/, backend/, server.js, and backup/.  
- Mention GitHub is connected and daily commits are pushed.  
- Explain how code is separated for clarity and maintenance.

## 🗄️ MySQL Database (Local)

- Open MySQL from Command Prompt.  
- Run:  
 USE user\_info;  
 SELECT \* FROM users;  
- Show the user record and explain that passwords are stored in plain text for learning purposes.

## 🌐 Frontend UI

- Open index.html from frontend folder.  
- Show the login form.  
- Explain validations: empty fields, email format, password length.  
- Mention how form feedback is shown via alerts and styled div.

## 🔧 Backend Server (Node.js)

- Run server using: node server.js  
- Show terminal log: 'Server started on http://localhost:3000'  
- Login request triggers: '🔍 Checking credentials for: user@example.com'  
- Explain backend checks credentials using MySQL and sends JSON response.

## ✅ Live Testing (Frontend + Backend + MySQL)

- Test 1: Correct email/password → shows alert 'Login successful!'  
- Test 2: Wrong password → shows message 'Invalid credentials'  
- Test 3: Empty fields → shows validation errors  
- Test 4: MySQL service down → shows 'Server error'

## 📦 GitHub Backup

- Show the latest commit message: 'Day-8: Completed login testing and error handling'  
- Explain backup strategy: GitHub + local folder (backup/frontend\_day8, backend\_day8)

## 🧠 Q&A – Common Interview/Demo Questions

* 1. Why did you skip password encryption (bcrypt)?

Because this is a learning-phase project focusing on basic login functionality. No sensitive data is involved.

* 2. Where is the database hosted?

Locally on the developer's machine using MySQL with the user\_info database.

* 3. How does the backend validate login?

It receives email/password via POST request, runs a MySQL SELECT query, and sends success/failure response.

* 4. How does the frontend handle errors?

Through input validation, on-screen messages, and catch blocks for backend failures.

* 5. How is the code backed up?

Code is version-controlled using Git and pushed to a GitHub repository daily. Local copies are stored in the backup folder.