IBM-NJ-USER AUTHENTICATION PHASE-5

**PROJECT NAME: USER AUTHENTICATION**

**TOPIC NAME: PROJECT DEMONSTRATION&DOCUMENTATTION**

**1.Final Demo Walkthrough**

**Step 1: Registration**

* Open the registration page.
* Enter username, email, and password.
* Password is encrypted (hashed) and stored in the database.
* A success message is shown after registration.

**Step 2: Login**

* Navigate to the login page.
* Enter registered email and password.
* System validates the credentials with the database.
* On success, the user is redirected to the dashboard.

**Step 3: Dashboard Access**

* Only authenticated users can access the dashboard.
* If an unauthenticated user tries to open it, they are redirected to the login page.

**Step 4: Logout**

* Click on the logout button.
* User session or token is cleared.
* User is redirected back to the login page.

**Step 5: Security Check (Optional)**

* Demonstrate that passwords are stored in encrypted form show that unauthorized access is blocked.

**2.Project Report-User Authentication**

**1. Introduction**

The *User Authentication System* is developed to ensure that only authorized users can access a specific application or system. Authentication is a crucial part of any software or web application, as it helps to protect user data and maintain the user authentication registration system can be  
This project demonstrates how a secure login and registration system can be implemented using modern technologies with password encryption, session management, and secure data handling.

**2. Objectives**

* To develop a secure user authentication mechanism.
* To allow users to register, login, and logout safely.
* To prevent unauthorized access to sensitive pages.
* To use password encryption for data security.
* To maintain session or token-based access control.

**3. System Requirements**

**Software Requirements:**

* Frontend: HTML, CSS, JavaScript (or React/Angular)
* Backend: Node.js / Python Flask / PHP (choose as per your project)
* Database: MySQL / MongoDB
* Tools: VS Code, XAMPP / Postman / Browser

**Hardware Requirements:**

* System with minimum 4 GB RAM
* Stable internet connection

**4. System Design and Workflow**

The authentication process follows a simple workflow:

1. **Registration:**
   * New users create an account by entering their details such as username, email, and password.
   * The password is hashed using a secure encryption algorithm before saving in the database.
2. **Login:**
   * Registered users enter their credentials.
   * The system verifies the entered data with the stored records.
   * If valid, the user is logged in and a session/token is generated.
3. **Dashboard / Secure Area:**
   * Only authenticated users can access this section.
   * If a non-logged-in user tries to access it, they are redirected to the login page.
4. **Logout:**
   * The user can log out, which destroys the session/token.
   * After logout, access to secure areas is blocked until the user logs in again.

**5. Security Features**

* **Password Hashing:** Ensures passwords are not stored as plain text.
* **Session/Token Management:** Maintains active user sessions securely.
* **Input Validation:** Prevents SQL injection and XSS attacks.
* **Access Restriction:** Only authenticated users can access protected routes.

**6. Results and Output**

The project successfully performs all key authentication operations:

* Users can register and login securely.
* Data is stored safely in the database.
* Unauthorized users are restricted from accessing private areas.
* Logout function clears the session properly.

Include screenshots in your report for:

* Registration Page
* Login Page
* Dashboard Page

**3.Screenshots/API Documentation**

To visually represent the system’s working,relevant screenshots have been included: A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect. **A screenshot of a computer

AI-generated content may be incorrect.**

The screenshot tell me about my project.

**5.Challenges/Solutions**

**1. Weak Passwords**

**Challenge:** Users often use simple or reused passwords, leading to easy hacking.  
**Solution:** Enforce strong password rules and use **Multi-Factor Authentication (MFA)** for extra security.

**2. Phishing Attacks**

**Challenge:** Hackers trick users to reveal login details through fake websites or emails.  
**Solution:** Use **Two-Factor Authentication (2FA)**, **Single Sign-On (SSO)**, and train users to identify fake links.

**3. Session Hijacking**

**Challenge:** Attackers steal session tokens to access accounts.  
**Solution:** Use **HTTPS**, **secure cookies**, and **session timeout** features.

**4. Biometric Spoofing**

**Challenge:** Fingerprint or face data can be faked or stolen.  
**Solution:** Add **liveness detection** and store biometric data securely in encrypted form.

**5. Usability vs. Security**

**Challenge:** Strong security can reduce user convenience.  
**Solution:** Use **adaptive authentication**—apply extra verification only in risky situations.

**6.GitHub README & Setup Guide**

**🧩 User Authentication System**

A simple and secure **user authentication** system using modern web technologies.  
Supports **user registration, login, logout, and session management** with encrypted passwords.

**⚙️ Features**

* User Registration & Login
* Password Hashing (using bcrypt)
* JSON Web Token (JWT) Authentication
* Protected Routes

**🛠️ Tech Stack**

* **Backend:** Node.js, Express.js
* **Database:** MongoDB
* **Auth:** JWT / bcrypt

**🚀 Setup Guide**

**1. Clone Repository**

git clone https://github.com/yourusername/user-authentication.git

cd user-authentication

**2. Install Dependencies**

npm install

**3. Create .env File**

PORT=5000

MONGO\_URI=your\_mongodb\_connection\_string

JWT\_SECRET=your\_secret\_key

**4. Run the Server**

npm start

**5. Test API Endpoints**

* **POST /register** → Create a new user
* **POST /login** → Authenticate and return JWT token
* **GET /profile** → Access protected route with token

**🔒 Security Features**

* Encrypted passwords
* Token-based authentication
* Secure session handling
* Security
* Overview secure

**8.Testing of Enhancements**

After implementing enhancements in the user authentication system, thorough testing is required to ensure security, reliability, and performance.

**1. Functional Testing:**

* Verify login, signup, password reset, and logout functions.
* Ensure multi-factor authentication (MFA) works correctly.

**2. Security Testing:**

* Check for vulnerabilities like SQL injection, brute-force attacks, and session hijacking.
* Validate data encryption during transmission and storage.

**3. Performance Testing:**

* Test response time under high user loads.
* Ensure the authentication server handles multiple requests efficiently.

**4. Usability Testing:**

* Confirm the authentication flow is user-friendly and easy to navigate.

**5. Regression Testing:**

* Ensure new enhancements don’t break existing functionality.

**9.Final Submission**

The final project submission includes:

**Repo+deployement link:**

**https://github.com/vinotha01/User--Authentication-.git**