# 🚀 Flask JWT Authentication Practice Assignment

## 1️⃣ Case Study: Secure Employee Portal for TechSolutions Inc.

TechSolutions Inc. wants to build a \*\*secure employee portal\*\* that allows employees to register, log in, and access a protected dashboard. To ensure security, authentication must be handled using \*\*JWT tokens stored in HTTP-only cookies\*\*.

\*\*Requirements:\*\*

* ✔️ Employees must be able to \*\*register\*\* with a username and password.
* ✔️ Employees must \*\*log in\*\* to receive a \*\*JWT token\*\*.
* ✔️ The JWT token must be \*\*stored securely in an HTTP-only cookie\*\*.
* ✔️ A \*\*protected dashboard\*\* should be accessible only after authentication.
* ✔️ Employees should be able to \*\*log out\*\*, which clears the JWT cookie.

## 2️⃣ Objectives

* By completing this assignment, you will learn to:
* ✔️ Implement user authentication using \*\*JWT and cookies\*\*.
* ✔️ Store \*\*user credentials securely\*\* using \*\*bcrypt\*\*.
* ✔️ Use \*\*Flask and Jinja2\*\* for rendering dynamic templates.
* ✔️ Apply \*\*OOPS concepts\*\* to manage user authentication.
* ✔️ Protect routes using \*\*JWT-based authentication\*\*.

## 3️⃣ Project Requirements

* Your Flask application must have the following functionalities:
* ✔️ \*\*User Registration\*\* - Allow new users to create accounts.
* ✔️ \*\*User Login\*\* - Authenticate users and generate a JWT token.
* ✔️ \*\*JWT Token Storage\*\* - Store JWT securely in HTTP-only cookies.
* ✔️ \*\*Protected Dashboard\*\* - Users can access this page only after login.
* ✔️ \*\*Logout\*\* - Clear JWT and end the session.