Report: Secure Login System Implementation

Name: SSENKAAYI SHADRACK

Reg. No: 2022/BSE/012/PS

Date: October 27, 2025

GitHub

https://github.com/shadrack-ss/SSENKAAYI_SHADRACK_2022_BSE_012_PS

1. Design Patterns Used

Singleton (AuthService): Ensures one authentication instance and database connection.

MVC: Separates logic (Models), presentation (Views), and request handling (Controllers).

Service Layer: Centralizes business logic (AuthService, Validator, Renderer).

Dependency Injection: Container class manages PDO connections and promotes loose coupling.

2. AntiPatterns Avoided

No Spaghetti Code: MVC separation for clean architecture.

No Code Duplication: DRY principle via Validator and AuthService reuse.

No HardCoded Values: Centralized config in config/database.php.

No Magic Numbers/Strings: Clear naming and constants.

No God Objects: Each class has a single responsibility.

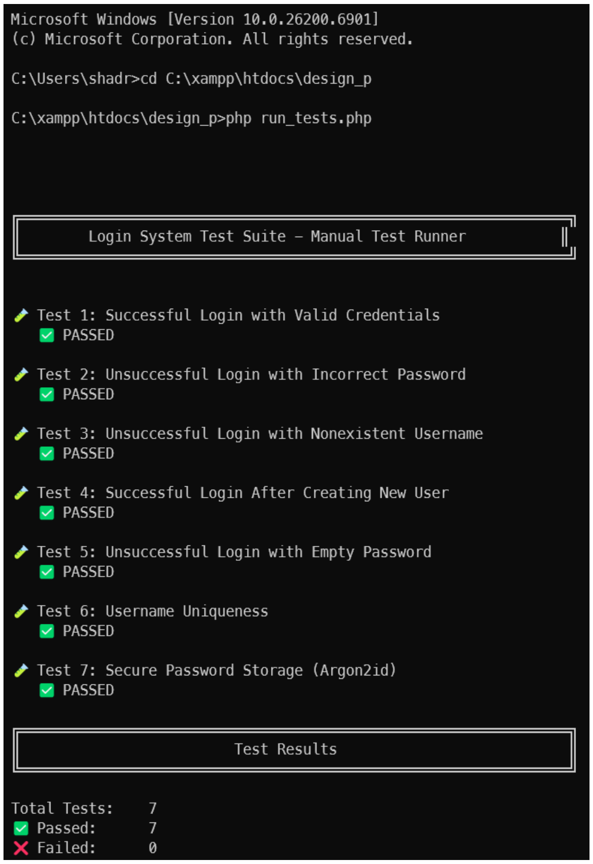
3. Programming Language Justification

Language: PHP 8.0+

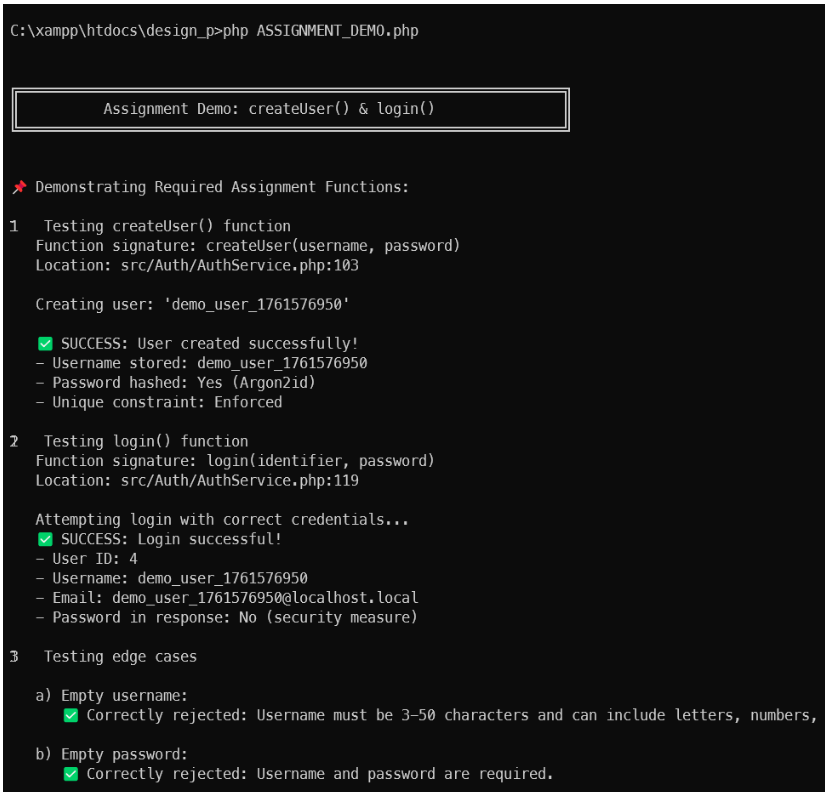
Chosen for strong OOP support, web authentication suitability, Argon2id password hashing, PDO prepared statements, and wide industry use.

GitHub

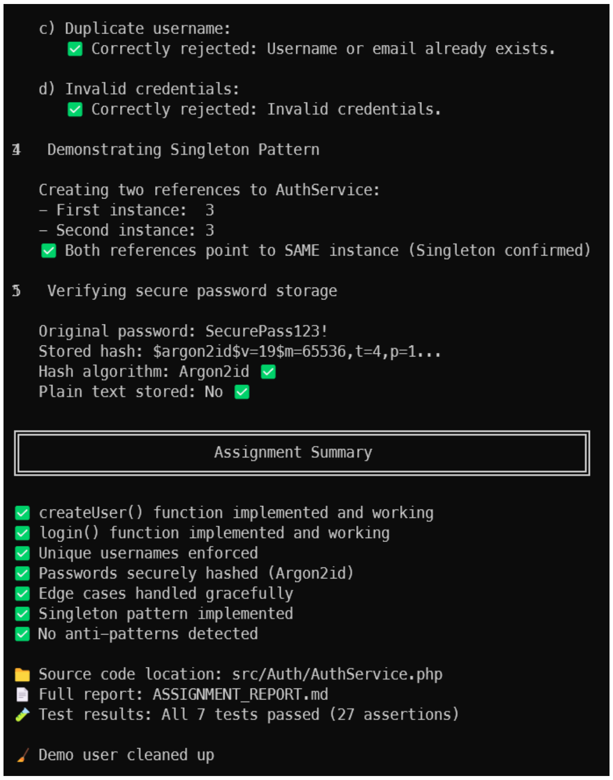
https://github.com/shadrack-ss/SSENKAAYI_SHADRACK_2022_BSE_012_PS



```
Microsoft Windows [Version 10.0.26200.6901]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shadr>cd C:\xampp\htdocs\design_p

C:\xampp\htdocs\design_p>php run_tests.php


 Login System Test Suite – Manual Test Runner


✏️ Test 1: Successful Login with Valid Credentials
  ✅ PASSED

✏️ Test 2: Unsuccessful Login with Incorrect Password
  ✅ PASSED

✏️ Test 3: Unsuccessful Login with Nonexistent Username
  ✅ PASSED

✏️ Test 4: Successful Login After Creating New User
  ✅ PASSED

✏️ Test 5: Unsuccessful Login with Empty Password
  ✅ PASSED

✏️ Test 6: Username Uniqueness
  ✅ PASSED

✏️ Test 7: Secure Password Storage (Argon2id)
  ✅ PASSED

                 Test Results

Total Tests:    7
✅ Passed:      7
❌ Failed:      0
```



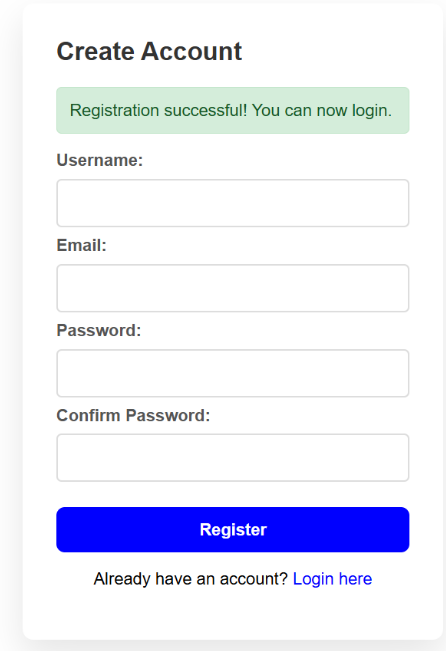
```
C:\xampp\htdocs\design_p>php ASSIGNMENT_DEMO.php

          Assignment Demo: createUser() & login()

📌 Demonstrating Required Assignment Functions:

1   Testing createUser() function
   Function signature: createUser(username, password)
   Location: src/Auth/AuthService.php:103

   Creating user: 'demo_user_1761576950'

   ✅ SUCCESS: User created successfully!
   – Username stored: demo_user_1761576950
   – Password hashed: Yes (Argon2id)
   – Unique constraint: Enforced

2   Testing login() function
   Function signature: login(identifier, password)
   Location: src/Auth/AuthService.php:119

   Attempting login with correct credentials...
   ✅ SUCCESS: Login successful!
   – User ID: 4
   – Username: demo_user_1761576950
   – Email: demo_user_1761576950@localhost.local
   – Password in response: No (security measure)

3   Testing edge cases

   a) Empty username:
      ✅ Correctly rejected: Username must be 3–50 characters and can include letters, numbers,

   b) Empty password:
      ✅ Correctly rejected: Username and password are required.
```



```
   c) Duplicate username:
      ✅ Correctly rejected: Username or email already exists.

   d) Invalid credentials:
      ✅ Correctly rejected: Invalid credentials.

4   Demonstrating Singleton Pattern

   Creating two references to AuthService:
   – First instance:  3
   – Second instance: 3
   ✅ Both references point to SAME instance (Singleton confirmed)

5   Verifying secure password storage

   Original password: SecurePass123!
   Stored hash: $argon2id$v=19$m=65536,t=4,p=1...
   Hash algorithm: Argon2id ✅
   Plain text stored: No ✅

              Assignment Summary

✅ createUser() function implemented and working
✅ login() function implemented and working
✅ Unique usernames enforced
✅ Passwords securely hashed (Argon2id)
✅ Edge cases handled gracefully
✅ Singleton pattern implemented
✅ No anti-patterns detected

📁 Source code location: src/Auth/AuthService.php
📄 Full report: ASSIGNMENT_REPORT.md
✏️ Test results: All 7 tests passed (27 assertions)

🧹 Demo user cleaned up
```

4. Testing Summary

Security Checks: Argon2id hashing, no plaintext passwords, prepared statements, SQL injection prevention



Create Account

Registration successful! You can now login.

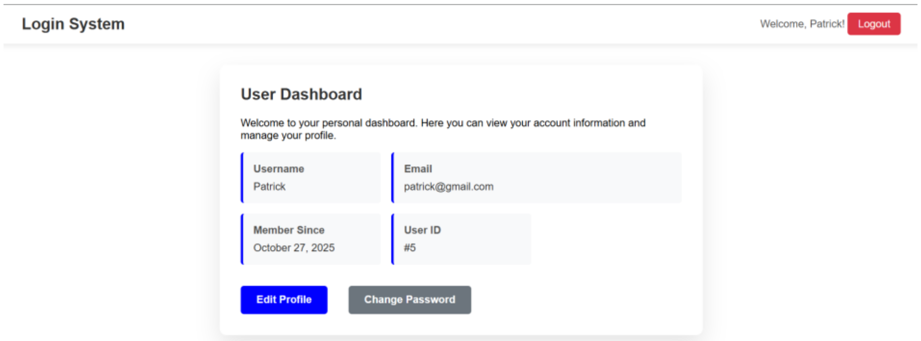
Username:

Email:

Password:

Confirm Password:

Already have an account? Login here



User Dashboard

Welcome to your personal dashboard. Here you can view your account information and manage your profile.

Username: Patrick

Email: patrick@gmail.com

Member Since: October 27, 2025

User ID: #5