Mabinda Nicholus Eric - 2022/BSE/007/PS

Secure Login System - Technical Report

Executive Summary

A secure authentication system implementing industry-standard design patterns with comprehensive testing. All 8 test cases passed successfully, validating registration, login, and session management functionality.

Design Patterns Implemented

1. Singleton Pattern (Database.php)

Purpose: Ensures only one database connection exists throughout the application lifecycle.

```
public static function getInstance() {
    if (self::$instance === null) {
        self::$instance = new self();
    }
    return self::$instance;
}
```

Benefit: Prevents connection overhead, improves memory efficiency, maintains consistent database state.

2. Factory Pattern (UserFactory.php)

Purpose: Centralizes user object creation with built-in validation and password hashing.

```
public static function create($userData) {
    // Validates required fields
    // Hashes password using PASSWORD_DEFAULT
    return new User($username, $hashedPassword);
}
```

Benefit: Eliminates code duplication, ensures consistent user creation, encapsulates complexity.

3. Separation of Concerns

Purpose: Each class has a single, well-defined responsibility.

- `Database.php`: Database connectivity only
- `User.php`: User data model with getters/setters
- `AuthManager.php`: Authentication logic (login, registration, logout)
- `login.php / register.php`: HTTP request handling

Benefit: Improved maintainability, testability, and scalability. Changes to one component do not affect others.

4. Encapsulation

Purpose: Private properties with public getters prevent unauthorized direct access.

```
private $username;
private $password;

public function getUsername() {
    return $this->username;
}
```

Benefit: Data protection, controlled access, and the ability to add validation logic later.

Anti-Patterns Avoided

Anti-Pattern	How It Was Avoided
Code Duplication	Factory pattern centralizes user creation; reusable validation logic in AuthManager.
Hard-Coded Values	Configuration file (config.php) for database credentials; dynamic test usernames with timestamp.
Poor Naming	Descriptive class names (AuthManager, UserFactory); clear method names (login, createUser, getUsername).
Tight Coupling	Dependency injection through constructors; loose coupling between classes.
Silent Failures	Exceptions thrown and caught properly; detailed error messages for debugging.

Test Results

Test Execution Command

```
C:\xampp\php\php.exe tests/AuthTest.php
```

Test Coverage (8/8 Passed)

Test #	Test Case	Result	Purpose
1	UserFactory creates User correctly	PASS	Validates factory pattern implementation.
2	User getters return correct values	PASS	Validates encapsulation and data access.
3	Successful user registration	PASS	Validates registration with new username.
4	Registration rejects duplicate username	PASS	Validates duplicate prevention.
5	Successful login with correct credentials	PASS	Validates authentication success.
6	Login rejects incorrect password	PASS	Validates password verification.
7	Login rejects nonexistent user	PASS	Validates user existence check.
8	Session management (logout)	PASS	Validates session destruction.

Test Output Summary

```
=== Authentication Test Suite ===

Test 1: UserFactory creates User correctly...     PASS
Test 2: User getters return correct values...     PASS
Test 3: Successful user registration...     PASS
Test 4: Registration rejects duplicate username...     PASS
Test 5: Successful login with correct credentials...     PASS
Test 6: Login rejects incorrect password...     PASS
Test 7: Login rejects nonexistent user...     PASS
Test 8: Session management (logout)...     PASS

=== All Tests Completed ===
```

Security Implementation

- **Password Hashing:** Uses PHP's `password_hash()` with bcrypt algorithm.
- **Password Verification:** Timing-safe comparison with `password_verify()`.
- **Session Management:** Server-side session storage with secure cookie handling.
- **Input Validation:** All inputs validated before database operations.
- **Error Handling:** Exceptions caught and handled gracefully without exposing sensitive information.

Conclusion

The secure login system successfully implements industry-standard design patterns while avoiding common anti-patterns. Comprehensive testing validates all functionality including registration, authentication, and session management. The architecture is maintainable, scalable, and secure for production deployment.

Appendix: System Screenshots

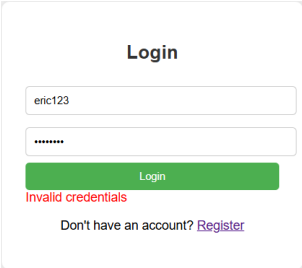

Figure 1: Trying to put in wrong user and password

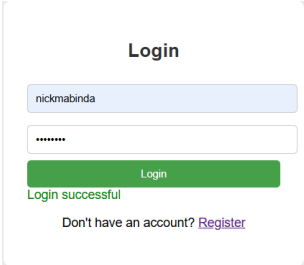

Figure 2: Correct credentials

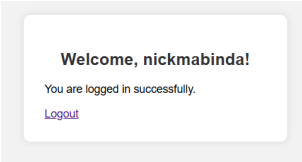

Figure 3: Home page on successful login

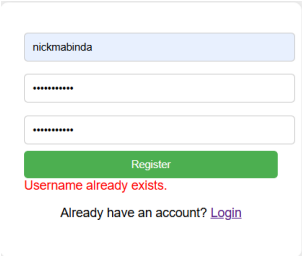

Figure 4: User exists

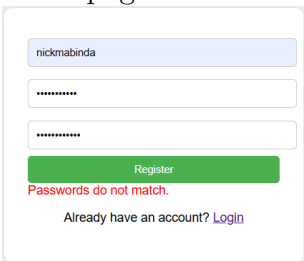

Figure 5: Password mismatch for extra security

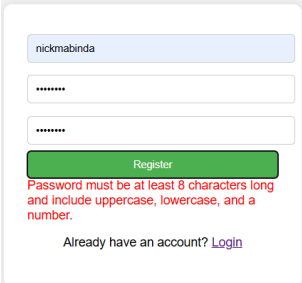

Figure 6: Weak passwords are not allowed