Team name: “QWERTY” LAB9

Members:

Danyil Tymchuk

Artem Surzhenko

Test Cases

Unit Testing

- Classes to test: `User`, `UserTest`, etc.
- Objects to update: any logic classes impacted by new requirements.
- Strategy: PHPUnit automation to cover methods like `getUsername()`, `getEmail()`, etc.

UI Testing

- Features: basic navigation, minimal clicks, and error recoverability.
- Enhancements: data-entry forms must provide clear test hooks for automated checks.

Requirements Testing

- Completed: core user functionality (displaying user info, database connectivity).
- Pending: additional form validation, user roles, and security checks.

Basis Path Test Calculations

- Determine the cyclomatic complexity per method (count decisions and loops).
- Plan test paths covering each branch (if conditions, loop boundaries).

Equivalence Partition Test Calculations

- Identify input ranges for user data (valid email format, username length, etc.).
- Group partitions (valid vs. invalid) and create representative tests.

Validation Tests

- Custom rules for fields (no blank usernames, valid email patterns, etc.).
- Verify error messages for malformed or missing inputs.

Unit Testing

- **Classes to Test**:
  - `User` class - For user management functionality
  - `Database` class - For database operations
  - `Validation` class - For input validation logic
  - `Authentication` class - For login/logout functionality

Objects to Update:

- Database connection objects need updates to accommodate transaction handling
- User object requires expansion to support additional profile fields
- Form handlers need additional validation methods

Testing Strategy:

- Create test cases for each public method in our classes
- Focus on testing boundary conditions and exception handling
- Implement mock objects to isolate units from external dependencies

UI Testing

- **Currently Implemented Features**:
  - Basic navigation structure
  - Form submission and feedback
  - Error message display
- **Features Needing Enhancement**:
  - Add consistent ID attributes to all interactive elements for test automation
  - Implement ARIA attributes for accessibility testing
  - Create consistent error handling patterns across all forms
  - Add data-testid attributes to critical UI components
  - Ensure all forms have proper tabindex attributes
- **Testing Focus Areas**:
  - Navigation efficiency (minimize clicks to complete common tasks)
  - Form completion time and error recovery
  - Responsive design across different viewport sizes
  - Browser compatibility (Chrome, Firefox, Safari)

Requirements Testing

- **Completed Requirements**:
  - User registration functionality
  - Basic authentication (login/logout)

    - User profile display
    - Database connectivity and CRUD operations
    - Basic security measures (password hashing)
- **Pending Requirements**:
    - Role-based access control
    - Advanced input validation
    - Password recovery workflow
    - User activity logging
    - Session timeout handling

Basis Path Test Calculations

1. **User Authentication Method**:
    - Control flow graph nodes: Entry → Check credentials → Validate input → Query database → Check password → Create session/Return error → Exit
    - Edges = 7, Nodes = 6
    - Cyclomatic complexity = E - N + 2 = 7 - 6 + 2 = 3
    - Independent paths to test:
        1. Valid credentials path
        2. Invalid username path
        3. Invalid password path
2. **User Registration Method**:
    - Control flow graph with validation branches increases complexity
    - Estimated cyclomatic complexity: 5
    - Key paths to test include duplicate email handling, password strength validation, and successful registration

Equivalence Partition Test Calculations

1. **Email Field**:
    - Valid partition: properly formatted emails (test@example.com)
    - Invalid partitions:
        - Missing @ symbol (testexample.com)
        - Missing domain (test@.com)
        - Invalid characters (!test@example.com)
        - Empty string ("")
2. **Password Field**:
    - Valid partition: 8+ characters with mixed case, numbers, and symbols
    - Invalid partitions:
        - Too short (less than 8 characters)
        - Missing uppercase letters
        - Missing numbers
        - Missing special characters

        - Common passwords ("password123")
3. **Age Field**:
    - Valid partition: 18-120
    - Invalid partitions:
        - Under 18
        - Over 120
        - Non-numeric input

Validation Tests

- **Username Validation**:
    - Must be 3-20 characters
    - Alphanumeric characters only (plus underscore and hyphen)
    - Must not start with a number
    - Cannot contain spaces
    - Must be unique in database
- **Email Validation**:
    - Must conform to RFC 5322 format
    - Domain must include valid TLD
    - Must be unique in database
    - Maximum length of 254 characters
- **Password Validation**:
    - Minimum 8 characters
    - At least one uppercase letter
    - At least one lowercase letter
    - At least one number
    - At least one special character
    - Cannot match username or email
    - Cannot be a common password (from known list)
- **Form Submission Validation**:
    - CSRF token verification
    - Honeypot fields to detect bots
    - Rate limiting to prevent brute force attempts
    - Input sanitization before processing

Each validation rule will have corresponding error messages that clearly explain the issue and how to resolve it. All validation will be performed both client-side (for immediate feedback) and server-side (for security).

Source Code:

GitHub Repository: https://github.com/DanyilT/WebDev-Project