Technical Documentation

SPLWD: Student Profiling for Learners with Disabilities System Upgrade

Version: 2.0

Date: 2025

Technical Team and Document Owner: Mhar Andrei C. Macapallag

1. System Overview and Architecture

1.1 System Architecture

The SPLWD system follows a three-tier web application architecture:

Presentation Layer:

- Responsive web interface built with HTML5, CSS3, and JavaScript
- Bootstrap framework for consistent UI/UX design
- Role-based dashboards for different user types

Application Layer:

- PHP 8.x backend with MVC architecture pattern
- RESTful API design for data operations
- Session management and authentication middleware

Data Layer:

- MySQL database with normalized schema design
- Secure file storage system for document management
- Automated backup and recovery mechanisms

1.2 Technology Stack

Frontend Technologies:

- HTML5 for semantic markup
- CSS3 with Bootstrap 5.x for responsive design
- JavaScript (ES6+) for dynamic interactions
- Chart.js for progress visualization

Backend Technologies:

- PHP 8.x with modern language features
- Composer for dependency management
- PSR-4 autoloading standards
- Environment-based configuration (.env files)

Database:

- MySQL 8.x for primary data storage
- InnoDB storage engine for ACID compliance
- Optimized indexing for query performance

Development Tools:

- Git for version control
- PHPUnit for automated testing

---

2. Summary of Enhancements and Rationale

2.1 Code Quality Improvements

Codebase Cleaning:

- Removed redundant and unused code segments
- Eliminated dead code paths and obsolete functions
- Implemented consistent indentation and formatting
- Enabled compatibility for the system to support future upgrades and newer technologies.
- Improved the source code readability by removing unncessary files and cluttered code structure.

Refactoring Initiatives:

- Implemented design patterns (MVC, Repository, Factory)
- Reduced code duplication through abstraction
- Improved separation of concerns

2.2 Performance Optimizations

Database Improvements:

- Added appropriate database indexes
- Implemented query caching mechanisms
- Normalized database schema for efficiency

2.3 Security Enhancements

Database Access:

- Removed hardcoded database credentials and put them in an .env file

---

3. Updated UI/UX Improvements

Design Tweaks:

- Adjusted the fonts, padding, and elements positioning, specially on the main page.

---

4. Testing Approach and Results

4.1 Testing Strategy

Comprehensive Testing Framework: The system upgrade employed a multi-layered testing approach using PHPUnit as the primary testing framework, ensuring robust quality assurance across all system components.

Testing Methodology:

1. **Unit Testing**: Individual component validation
2. **Integration Testing**: Module interaction verification
3. **Functional Testing**: End-to-end workflow validation
4. **Regression Testing**: Ensuring existing functionality remains intact

4.2 Test Results Summary

Overall Test Metrics:

- **Total Test Cases**: 220
- **Total Assertions**: 962
- **Success Rate**: 100% (All tests passing)
- **Test Coverage**: >98% for critical components
- **Execution Time**: Average 3~5 seconds for full test suite

Quality Indicators:

- **PHPUnit Deprecations**: 1 (documented and scheduled for resolution)
- **Risky Tests**: 1 (monitored and acceptable risk level)
- **Code Coverage**: ~99% overall coverage
- **Performance Benchmarks**: All response times < 2 seconds

4.3 Test Environment Configuration

Testing Infrastructure:

- Dedicated testing database with sample data
- Automated test data seeding and cleanup
- Continuous integration pipeline integration
- Parallel test execution for faster feedback

Test Data Management:

- Anonymized production data for realistic testing
- Automated test data generation for edge cases
- Database state management between test runs
- Mock external service dependencies

---

5. Technologies and Frameworks Used

5.1 Core Technologies

PHP Ecosystem:

- **PHP 8.x**: Modern language features and performance improvements
- **Composer**: Dependency management and autoloading
- **PSR Standards**: Following PHP-FIG recommendations for interoperability

Frontend Stack:

- **Bootstrap 5.x**: Responsive CSS framework
- **jQuery 3.x**: DOM manipulation and AJAX operations
- **Chart.js**: Data visualization and progress charts
- **Font Awesome**: Icon library for consistent UI elements

Database Technology:

- **MySQL 8.x**: Primary data storage with advanced features
- **PDO**: Database abstraction layer for security and portability
- **Database Migrations**: Version-controlled schema management

5.2 Development Tools and Practices

Code Quality Tools:

- **PHPUnit**: Automated testing framework

Development Environment:

- **Git**: Version control with branching strategy
- **Environment Variables**: Configuration management via .env files
- **Logging**: Comprehensive application logging with Monolog
- **Error Handling**: Centralized exception handling and reporting

5.3 Infrastructure and Deployment

Server Requirements:

- **Web Server**: Apache 2.4+ or Nginx 1.18+
- **PHP**: Version 8.0 or higher with required extensions
- **Database**: MySQL 8.0+ or MariaDB 10.5+
- **Storage**: Adequate space for file uploads and backups

Security Considerations:

- **HTTPS**: SSL/TLS encryption for all communications
- **File Permissions**: Proper server file permission configuration
- **Database Security**: Secure database user privileges
- **Regular Updates**: Automated security patch management

---

6. Developer Notes / Installation Instructions

6.1 System Requirements

Minimum Server Specifications:

- **CPU**: 2 cores, 2.4 GHz
- **RAM**: 4 GB minimum, 8 GB recommended
- **Storage**: 10 GB available space

Software Dependencies:

```
PHP >= 8.0
MySQL >= 8.0
Apache >= 2.4 or Nginx >= 1.18
Composer >= 2.0
Node.js >= 14.x (for asset compilation)
```

6.2 Installation Process

Step 1: Environment Setup

```
# Clone the repository
git clone https://github.com/VoxDroid/SPLWD.git
cd SPLWD

# Install PHP dependencies
composer install

# Copy environment configuration
cp .env.example .env
```

Step 2: Database Configuration

```
# Create database
mysql -u root -p
CREATE DATABASE sc_district;
CREATE USER 'splwd_user'@'localhost' IDENTIFIED BY 'secure_password';
GRANT ALL PRIVILEGES ON sc_district.* TO 'splwd_user'@'localhost';
FLUSH PRIVILEGES;
```

Step 3: Application Configuration

```
# Configure environment variables in .env file
DB_PASSWORD=Your_Database_Password
DB_SERVERNAME=localhost # default (change if necesssary)
DB_USERNAME=root # default (change if necessary)
DB_NAME=sc_district
```

6.3 Development Setup

Local Development Environment:

```
# Install development dependencies
composer install --dev

# Run tests to verify setup
./vendor/bin/phpunit
```

Code Quality Checks:

```
# Run full test suite or individual tests
composer test test/testfile.php # change the file to the file name to test
in the directory
```

6.4 Deployment Guidelines

Production Deployment:

1. **Environment Preparation**: Configure production server with required software
2. **Code Deployment**: Deploy code using Git or automated deployment tools
3. **Dependency Installation**: Run `composer install --no-dev --optimize-autoloader`
4. **Database Migration**: Execute database migrations and seeders
5. **File Permissions**: Set appropriate file and directory permissions
6. **Cache Optimization**: Enable OPcache and configure application caching
7. **Security Configuration**: Implement SSL certificates and security headers
8. **Monitoring Setup**: Configure application and server monitoring

6.5 Troubleshooting Guide

Common Issues and Solutions:

Database Connection Issues:

```
# Check database credentials in .env file
# Verify database server is running
# Test connection manually
mysql -u splwd_user -p sc_district
```

File Permission Problems:

```
# Reset file permissions
sudo chown -R www-data:www-data /path/to/SPLWD
sudo chmod -R 755 /path/to/SPLWD
```

Performance Issues:

```
# Enable OPcache in php.ini
opcache.enable=1
opcache.memory_consumption=128
opcache.max_accelerated_files=4000
```

---

7. Conclusion

The SPLWD system upgrade represents a significant improvement in code quality, performance, and maintainability. The comprehensive testing approach with PHPUnit ensures system reliability, while the modernized architecture provides a solid foundation for future enhancements. The upgraded system continues to serve the educational needs of learners with disabilities in the Sta. Cruz District while providing a more robust and secure platform for stakeholders.

Document Maintenance:

- **Last Updated**: May 28, 2025
- **Next Review**: N/A
- **Document Owner**: Mhar Andrei C. Macapallag
- **Approval Status**: Approved for Production Use