Software Requirements Specification (SRS) for Document Tracking System

---

1. Introduction and Purpose

1.1 Purpose

This Software Requirements Specification (SRS) document describes the requirements for the Document Tracking System developed for the National Irrigation Administration Region IV-A (CALABARZON) in Pila, Laguna. The system is designed to track, manage, and monitor documents as they move through various offices and departments within the organization.

1.2 Scope

The Document Tracking System provides a comprehensive solution for managing document workflows, from creation and submission to processing, transfer, and completion. It enables users to track documents in real-time, receive notifications about document status changes, and maintain a complete history of document movements and actions taken.

1.3 Intended Audience

This SRS is intended for:

- System developers and maintainers
- System administrators
- End users (administrators, record office staff, handlers, and guests)
- Project stakeholders and management

1.4 Definitions and Acronyms

- **DTS**: Document Tracking System
- **QR Code**: Quick Response Code used for document identification
- **Handler**: User responsible for processing documents within a specific office
- **Record Office**: Central office responsible for document management (Administrative Section Records)
- **Guest**: External user who can submit documents to the system

---

2. Overall Description

2.1 System Overview

The Document Tracking System is a web-based application that facilitates the tracking and management of documents as they move through various offices within the National Irrigation Administration. The system uses a centralized database to store document information, user data, and tracking history. It implements QR code technology for easy document identification and tracking.

2.2 System Architecture

The system follows a traditional web application architecture with:

- **Frontend**: HTML, CSS, JavaScript, Bootstrap for responsive design
- **Backend**: PHP for server-side processing
- **Database**: MySQL (MariaDB) for data storage
- **Additional Libraries**: PHPMailer for email notifications, phpqrcode for QR code generation

2.3 User Classes and Characteristics

2.3.1 System Administrator (sysadmin)

- Manages user accounts and access rights
- Configures system settings
- Monitors system performance
- Manages office information

2.3.2 Administrator (admin)

- Manages document types and categories
- Oversees document workflows
- Generates reports
- Manages office names and codes

2.3.3 Record Office Staff (Administrative Section Records)

- Receives and processes incoming documents
- Assigns tracking codes to documents
- Transfers documents to appropriate offices
- Monitors document status

2.3.4 Document Handler

- Receives and processes documents within their office
- Updates document status and adds action taken
- Transfers documents to other offices
- Completes document processing

2.3.5 Guest

- Submits documents to the system
- Tracks the status of submitted documents
- Receives notifications about document status changes

2.4 Operating Environment

- **Web Server**: Apache/Nginx
- **Database Server**: MySQL/MariaDB
- **Client**: Modern web browsers (Chrome, Firefox, Safari, Edge)
- **Development Environment**: Laragon (local development environment)

---

3. Functional and Non-functional Requirements

3.1 Functional Requirements

3.1.1 User Authentication and Authorization

- The system shall provide secure login functionality for all users.
- The system shall support different user roles with specific permissions.
- The system shall allow users to reset forgotten passwords via email.
- The system shall enforce password security policies.
- The system shall maintain user session information.

3.1.2 Document Management

- The system shall allow users to upload new documents with metadata.
- The system shall generate unique tracking codes for documents.
- The system shall generate QR codes for easy document identification.
- The system shall maintain document version history.
- The system shall support various document types and categories.

- The system shall allow document search and filtering.

3.1.3 Document Tracking

- The system shall track document movement between offices.
- The system shall record all actions taken on documents.
- The system shall maintain a complete history of document processing.
- The system shall allow users to view the current status of any document.
- The system shall support QR code scanning for quick document lookup.

3.1.4 Workflow Management

- The system shall support document transfer between offices.
- The system shall allow users to add comments and action notes to documents.
- The system shall support document completion and archiving.
- The system shall allow document rejection with reasons.

3.1.5 Notification System

- The system shall notify users about new documents assigned to them.
- The system shall notify users about document status changes.
- The system shall alert users about idle documents requiring attention.
- The system shall support email notifications for critical events.
- The system shall maintain a notification history for each user.

3.1.6 Reporting

- The system shall generate reports on document processing statistics.
- The system shall provide dashboards with document status summaries.
- The system shall allow export of reports in various formats.

3.1.7 User Management

- The system shall allow administrators to create and manage user accounts.
- The system shall support user registration with approval workflow.
- The system shall allow users to update their profile information.
- The system shall support user account deactivation and archiving.

3.1.8 Additional Enhancements

3.1.8.1 Modular Code Refactoring

The system now includes modular PHP classes (**BaseModel**, **DocumentType**, **ActionTaken**, etc.) to improve code reusability and reduce redundancy.

3.1.8.2 Secure Configuration Handling

A global **Config** class was added to manage application-wide settings such as time zones and email credentials.

3.1.8.3 Database Connection Abstraction

A singleton-based **Database** class was introduced to manage database connections securely and efficiently.

3.1.8.4 Automated Testing

Unit tests were written using PHPUnit and integrated with GitHub Actions to automate testing on every code update.

3.1.8.5 Database Optimization & Archiving

Scripts were developed to:

- Archive old notifications and completed documents
- Optimize MySQL tables
- Detect and alert on idle documents

This ensures long-term performance and data hygiene.

3.1.8.6 Guest Dashboard Interface

A new dashboard was created specifically for Guest users to:

- Track document status
- View processing history

3.2 Non-functional Requirements

3.2.1 Performance

- The system shall load pages within 3 seconds under normal load conditions.
- The system shall support concurrent access by multiple users.
- The system shall handle at least 1000 document transactions per day.

3.2.2 Security

- The system shall encrypt all passwords stored in the database.

- The system shall implement secure session management.
- The system shall validate all user inputs to prevent injection attacks.
- The system shall maintain audit logs of critical actions.
- The system shall enforce role-based access control.

3.2.3 Reliability

- The system shall be available 99% of the time during working hours.
- The system shall include error handling mechanisms.
- The system shall maintain data integrity during concurrent operations.

3.2.4 Usability

- The system shall have a responsive design for various screen sizes.
- The system shall provide intuitive navigation and user interfaces.
- The system shall include help documentation for users.
- The system shall provide meaningful error messages.

3.2.5 Maintainability

- The system shall follow a modular architecture for easy maintenance.
- The system shall include database optimization tools.
- The system shall support configuration changes without code modification.

3.2.6 Testing Tools and Automation

This system incorporates a suite of modern testing and automation tools to ensure high-quality releases and stable code:

- PHPUnit is used for writing and running unit tests for core business logic, such as document tracking, user roles, and database operations.
- Composer manages dependencies and enables autoloading of PHP classes, allowing easy test execution and future extensibility.
- GitHub Actions automates the execution of unit tests on every push to the repository, ensuring that any regression is caught early.

---

4. System Features and Interfaces

4.1 User Interfaces

4.1.1 Login and Registration

- Login page with username and password fields
- Registration page for new users
- Password recovery interface
- OTP verification page for secure registration

4.1.2 Dashboards

- Role-specific dashboards showing relevant information
- Document status summaries and counts
- Recent activity logs
- Notification indicators

4.1.3 Document Management Interfaces

- Document upload form with metadata fields
- Document search and filter interface
- Document details view with history and tracking information
- Document action forms (transfer, complete, comment)

4.1.4 User Management Interfaces

- User list with filtering options
- User creation and editing forms
- User profile management
- Password change interface

4.1.5 QR Code Interfaces

- QR code generation for documents
- QR code scanning interface
- QR code printing layout

4.2 Hardware Interfaces

- Camera support for QR code scanning
- Printer support for document and QR code printing

4.3 Software Interfaces

- Database management system (MySQL/MariaDB)
- Email server for notifications
- Web server (Apache/Nginx)

4.4 Communication Interfaces

- HTTP/HTTPS for web communication
- SMTP for email notifications

---

5. Assumptions and Constraints

5.1 Assumptions

- Users have basic computer literacy and can navigate web applications.
- Users have access to internet-connected devices with modern web browsers.
- The organization has a stable network infrastructure.
- Document handlers are assigned to specific offices.
- Each office has at least one designated document handler.
- The system will primarily be used during regular office hours.
- Email notifications are an acceptable form of communication for users.

5.2 Constraints

5.2.1 Technical Constraints

- The system must be developed using PHP and MySQL.
- The system must work with existing hardware infrastructure.
- The system must support common web browsers.
- The system must operate within the organization's network security policies.

5.2.2 Business Constraints

- The system must comply with organizational document handling policies.
- The system must maintain document confidentiality and integrity.
- The system must support the existing organizational structure.

5.2.3 Regulatory Constraints

- The system must comply with relevant data protection regulations.

- The system must maintain audit trails for accountability.

---

6. Use Case Diagrams or Descriptions

6.1 Document Submission and Initial Processing

Actors: Guest, Record Office Staff

Description: A guest submits a document to the system. The record office staff reviews the document, assigns a tracking code, and transfers it to the appropriate office.

Flow:

1. Guest logs into the system.
2. Guest uploads a document with required metadata.
3. System notifies record office staff about the new document.
4. Record office staff reviews the document.
5. Record office staff assigns a tracking code and generates a QR code.
6. Record office staff transfers the document to the appropriate office.
7. System notifies the receiving office about the incoming document.
8. System updates the document status and history.

6.2 Document Processing by Handler

Actors: Document Handler

Description: A document handler receives a document, processes it, adds action notes, and either completes it or transfers it to another office.

Flow:

1. Handler logs into the system.
2. Handler views incoming documents in their dashboard.
3. Handler selects a document to process.
4. Handler reviews the document details.
5. Handler adds action notes or comments.
6. Handler decides to either: a. Complete the document processing. b. Transfer the document to another office.
7. System updates the document status and history.

8. System sends notifications to relevant users.

6.3 Document Tracking

Actors: Any User

Description: A user tracks the status and history of a document using its tracking code or QR code.

Flow:

1. User logs into the system.
2. User either: a. Enters the document tracking code. b. Scans the document QR code.
3. System displays the document details, current status, and history.
4. User views the document's movement through different offices.
5. User views actions taken on the document.

6.4 User Management

Actors: System Administrator, Administrator

Description: An administrator manages user accounts, including creation, modification, and deactivation.

Flow:

1. Administrator logs into the system.
2. Administrator navigates to the user management section.
3. Administrator performs one of the following actions: a. Views the list of users. b. Creates a new user account. c. Modifies an existing user's details. d. Activates or deactivates a user account.
4. System updates the user database.
5. System notifies affected users about account changes.

6.5 Notification Management

Actors: Any User

Description: A user receives and manages notifications about document-related events.

Flow:

1. User logs into the system.
2. System displays unread notifications in the user's dashboard.

3. User views notification details.
4. User marks notifications as read.
5. User takes action based on notification content (e.g., views a document).

6.6 Report Generation

Actors: Administrator, Record Office Staff

Description: An authorized user generates reports about document processing statistics.

Flow:

1. User logs into the system.
2. User navigates to the reporting section.
3. User selects report type and parameters (date range, document type, etc.).
4. System generates the requested report.
5. User views the report on screen or exports it in the desired format.