EUROPEAN UNION

unicef
for every child

SEMIS
TECHNICAL USER GUIDE

About this document

Recommended citation:
Sindh Education Management Information System Technical User Manual / Operational Guide 2023

This is a living document and changes will be made as per the current or updated system.

TABLE OF CONTENTS

REVISION HISTORY

Name	Date	Reason for Changes	Version

LIST OF ACRONYMS

API	Application Programming Interface
APP	Application
CSS	Cascading Style Sheets
DBMS	Database Management System
HTML	Hypertext Markup Language
ORM	Object Relational Mapping
RDBMS	Relational Database Management System
SDLC	Software Development Lifecycle
SEMIS	School Education & Management Information System

CHAPTER 1

INTRODUCTION

INTRODUCTION OF XPERTFRAMEWORK:

XpertFramework, an open-source web application development framework, is designed to simplify the creation, customization, and deployment of web applications. Developed in Python and JavaScript, XpertFramework follows a full-stack model, providing both the backend and frontend components needed for building modern, feature-rich applications. Its design philosophy centers around simplicity, modularity, and extensibility, making it an excellent choice for a wide range of applications, from small business tools to large-scale enterprise solutions.

XpertFramework adopts a modular architecture that allows developers to build applications as a collection of interconnected modules. Each module represents a functional unit, and these modules can be easily integrated or extended, providing flexibility and scalability. The framework encourages code reusability and maintainability through the use of modular components.

One of the distinctive features of XpertFramework is its use of a document-oriented database, with each data record treated as a document. This approach simplifies data modeling and allows for dynamic schema changes without the need for complex migrations. Xpert's database engine, powered by SQLite or MariaDB, provides a reliable foundation for data storage and retrieval.

XpertFramework includes a sophisticated web forms system that allows developers to define data entry forms with minimal effort. The framework comes with a variety of built-in UI components, making it easy to create interactive and user-friendly interfaces. Xpert's client-side scripting capabilities, based on JavaScript and jQuery, enable dynamic and responsive user experiences.

PURPOSE OF THE SYSTEM ADMINISTRATOR MANUAL:

The purpose of this System Administrator Manual is to guide system administrators through the setup, maintenance, and troubleshooting of a XpertFramework-based system. Whether you are responsible for a small business deployment or managing a large-scale enterprise instance, this manual aims to provide comprehensive insights into the various aspects of system administration, ensuring the efficient and secure operation of your Xpert-based applications.

TARGET AUDIENCE:

This manual is designed for system administrators, DevOps engineers, and IT professionals who are involved in the deployment, configuration, and maintenance of XpertFramework-based systems. It assumes a basic understanding of system administration concepts, including server management, database administration, and web server configuration. Whether you are a seasoned administrator or new to XpertFramework, this guide will equip you with the knowledge and tools necessary to effectively manage your Xpert-based applications.

Throughout this manual, you will find step-by-step instructions, best practices, and valuable tips to optimize the performance, security, and reliability of your XpertFramework deployment. As the XpertFramework evolves, this manual will be updated to reflect the latest practices and features, ensuring that you have access to relevant and accurate information.

Now, let's embark on the journey of mastering the administration of XpertFramework-based systems, empowering you to build, maintain, and troubleshoot robust web applications with confidence.

CHAPTER 2

SYSTEM REQUIREMENTS

HARDWARE SPECIFICATIONS

Before installing and configuring an XpertFramework-based system, it is essential to ensure that your hardware meets the recommended specifications for optimal performance. The specific requirements may vary based on factors such as the size of your organization, the number of concurrent users, and the complexity of your applications. However, the following are general hardware guidelines.

Processor: Multi-core processor with a clock speed suitable for the expected load.

RAM: At least 4GB of RAM, with additional memory for larger deployments.

Storage: Sufficient disk space for the operating system, XpertFramework, and application data. Consider using SSDs for improved performance.

SOFTWARE PREREQUISITES:

XpertFramework relies on a specific set of software components to function correctly. Ensure that the following software prerequisites are installed on your server before proceeding with the installation.

Operating System: XpertFramework is compatible with Linux distributions such as Ubuntu, Debian, and CentOS. Choose a version that is supported by the specific version of Xpert you are installing.

Python: XpertFramework is written in Python, and a compatible version (commonly Python 3.x) must be installed on the server.

Database Server: XpertFramework supports databases like MariaDB and MySQL. Choose and install a compatible database server based on your preferences and requirements.

SUPPORTED OPERATING SYSTEMS AND DATABASES:

XpertFramework is designed to be versatile, supporting various combinations of operating systems and databases. However, it is essential to refer to the official documentation or release notes for the specific version you are installing to ensure compatibility.

Operating Systems: Commonly used Linux distributions such as Ubuntu 18.04/20.04, Debian 10, and CentOS 7/8 are well-supported. Refer to the official documentation for any version-specific recommendations.

Databases: MariaDB and MySQL are widely supported by Frappe. Ensure that you install and configure the preferred database server before initiating the Frappe installation process.

PRE-INSTALLATION CONSIDERATION:

Before starting the installation process, consider the following pre-installation tasks:

Network Configuration: Ensure that the server has a stable internet connection, as the installation process may involve downloading dependencies from the internet.

Firewall Settings: Adjust firewall settings to allow incoming connections to the required ports (e.g., HTTP, HTTPS, and database ports).

DNS Configuration: Set up appropriate domain names and ensure that DNS records are configured correctly if you plan to use domain-based access.

CHAPTER 3

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

XPERTFRAMEWORK IN SDLC:

1. The XpertFramework is designed to facilitate rapid development and customization throughout the entire SDLC.
2. XpertFramework's modular architecture supports iterative development, allowing for continuous enhancements and updates.
3. The integrated nature of XpertFramework simplifies testing and deployment processes.

ITERATIVE DEVELOPMENT:

1. Emphasis on iterative development cycles, enabling the incorporation of feedback and changes.
2. XpertFramework's document-oriented approach allows for flexible modifications and additions during development.

VERSION CONTROL:

1 Implementation of version control practices using tools like Git to track changes and manage collaborative development.
2 Branching and merging strategies for effective collaboration among development teams.

CONTINUOUS INTEGRATION/CONTINUOUS DEPLOYMENT (CI/CD):

1 Integration of CI/CD pipelines to automate testing and deployment processes.
2 Ensuring a streamlined and efficient workflow from development to production.

BEST PRACTIES:

Documentation:

1 Comprehensive documentation at each SDLC phase, including design documents, user manuals, and release notes.

2 Utilization of XpertFramework's built-in documentation features for code comments and API documentation.

Collaboration and Communication:

1 Effective communication channels and collaboration tools for project teams.

2 Regular status meetings, sprint reviews, and feedback sessions to ensure alignment with project goals.

CHAPTER 4

ARCHITECTURE

OVERVIEW OF SYSTEM ARCHITECTURE:

XpertFramework is a full-stack web based framework and it includes all the tools needed to deploy a site into production. Database, caching, background jobs, realtime notifications, etc are all configured when you set up a XpertFramework site.

XpertFramework is based on Python, so it uses the virtualenv to setup isolated environments for multiple Python versions. You can also use it to deploy sites with different XpertFramework versions.

App Architecture

1. Modular structure based on XpertFramework apps.
2. Each app encapsulates specific features and functionalities.
3. Apps can be custom or built on existing XpertFramework modules.

COMMUNICATION:

Client-Server Interaction:

1. Browser-based client interacting with the XpertFramework server.
2. AJAX requests seamless user interactions.
3. Web Sockets for real-time updates.

APIs and Integrations:

1. RESTful APIs for external integrations.
2. Integration with third-party services using XpertFramework's connectors.

KEY COMPONENTS:

Web Interface:

1. User interacts with the application through a web-based interface.
2. Built using XpertFramework's Desk framework for a responsive and intuitive experience.

XpertFramework Server:

1. Manages HTTP requests from the client.
2. Processes business logic and interacts with the database.

Database Layer:

1. Utilizes a relational database (e.g., MariaDB) to store data.
2. XpertFramework ORM for seamless interaction with the database.

Background Jobs:

1. Executes asynchronous tasks and background jobs.
2. Ensures scalability and responsiveness.

File System:

1. Stores file attachments and media assets.
2. Integration with the XpertFramework for efficient file management.

KEY ARCHITECTURAL DECISIONS:

Documentation- Oriented Architecture:

1. XpertFramework's document-oriented approach allows for a flexible and extensible data model.
2. Each entity is represented as a document, making it easy to customize and extend.

XpertFramework Apps:

1. Leveraging the modular structure of XpertFramework apps for better organization and maintainability.
2. Custom apps for specific business functionalities.

WebSockets for Real-Time Updates:

1. Real-time communication between the server and clients using WebSockets.
2. Enables instant updates and notifications for users.

CHAPTER 5

TECHNOLOGY STACK

OVERVIEW OF SYSTEM ARCHITECTURE:

The technology stack of the comprises a set of tools, frameworks, and libraries carefully chosen to build a robust and scalable system. In this chapter, we will explore the key components of the technology stack, providing insights into their roles and contributions to the overall functionality of the application.

PROGRAMMING LANGUAGES:

Python:

1. Core language for XpertFramework development.
2. Known for its simplicity and readability.
3. Widely used for backend logic and business rule implementation.

JavaScript:

1. RESTful APIs for external integrations.
2. Integration with third-party services using XpertFramework's connectors.

XPERTFRAME:

Web Interface:

1. Built on top of Python and follows the Flask web framework.
2. Facilitates rapid development and customization through its modular and extensible architecture.

ORM (Object-Relational Mapping)::

1. XpertFramework ORM simplifies database interactions, allowing developers to work with Python objects instead of SQL queries.

2. Ensures a seamless mapping between the application's data model and the underlying database.

Restful API:

1. Enables easy integration with external systems and services.
2. Supports CRUD (Create, Read, Update, Delete) operations over HTTP.

FRONTEND TECHNOLOGIES:

HTML and CSS:

1. Fundamental building blocks for web page structure and styling.
2. XpertFramework Desk uses Jinja templating for dynamic content generation.

JavaScript Libraries (e.g., jQuery):

1. Enhances client-side scripting and user interactions.
2. XpertFramework Desk leverages jQuery for DOM manipulation and AJAX requests.

DATABASE:

MariaDB:

1. XpertFramework utilizes a relational database to store application data.
2. Ensures data integrity, consistency, and scalability.

Additional Technologies:

1. Redis:

1. In-memory data structure store used for caching and improving performance.
2. Supports key-value storage and data structures.

2. WebSockets:

1. Facilitates real-time communication between the server and clients.
2. Used for instant updates and notifications within the XpertFramework Desk.

PYTHON AND XPERTFRAMEWORK:

Rapid Development:

1. Python's simplicity accelerates development cycles.
2. XpertFramework conventions reduce boilerplate code, fostering rapid application development.

MariaDB:

1. Relational Model:

1. MariaDB provides a robust and scalable relational database solution.
2. Well-suited for applications with complex data relationships.

2.Redis

Caching and Performance:

1. Redis enhances system performance through efficient caching.
2. Enables quick retrieval of frequently accessed data.

CHAPTER 6

DATABASE ARCHITECTURE

OVERVIEW OF DATABASE ARCHITECTURE:

Database architecture is a crucial component of any software system, determining how data is organized, stored, and accessed. In the context of Frappe Framework, the architecture revolves around the design and management of databases that store information for the applications. Below is an exploration of key elements in the database architecture for a Frappe-based system.

DATABASE MANAGEMENT SYSTEM (DBMS):

The choice of a DBMS plays a pivotal role in the database architecture. XpertFramework primarily supports both relational and NoSQL databases, offering flexibility based on the specific requirements of the application.

RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS):

XpertFramework often utilizes RDBMS like MariaDB or MySQL for structured data storage. This type of database organizes data into tables with predefined relationships, ensuring data integrity through the enforcement of constraints.

DOCUMENT-ORIENTED DATABASE STRUCTURE:

XpertFramework employs a document-oriented database structure, treating each data record as a document. This architecture is particularly prominent when using the NoSQL approach. Each document, typically represented in JSON format, encapsulates information related to an entity (e.g., a user, a transaction).

DATABASE SCHEMA:

The database schema defines the structure of the database, specifying how data is organized into tables, fields, and relationships. XpertFramework dynamic nature allows for the creation of custom doctypes, essentially custom tables, which enables high adaptability to various application needs.

Core Tables:

1. **`tabDocType`:**
   - Stores metadata about document types in the system.

- Defines fields, permissions, and behavior for each document.

2. **`tabDocField`:**
   - Contains information about fields associated with document types.
   - Specifies field properties such as type, label, and options.
3. **`tabDoTable`:**
   - Defines the relationships between different document types.
   - Represents the structure of documents in a tabular format.

ORM (OBJECT-RELATIONAL MAPPING):

XpertFramework incorporates an ORM layer that facilitates communication between the application and the database. This abstraction allows developers to interact with the database using high-level programming constructs in Python, without delving into SQL intricacies. SQLAlchemy is an example of an ORM used within XpertFramework.

DATA RELATIONSHIPS AND LINK FIELDS:

One of XpertFramework distinguishing features is its ability to define and manage data relationships easily. Link fields establish connections between different documents, creating relationships such as one-to-many or many-to-many. This flexibility is crucial for building comprehensive applications, especially in the context of ERP systems.

INDEXES AND PERFORMANCE OPTIMIZATION:

Efficient database performance is achieved through the strategic use of indexes. XpertFramework administrators can configure indexes on specific fields to expedite data retrieval. Proper indexing is essential for optimizing query performance, particularly in applications dealing with large datasets.

DATABASE CACHING:

To enhance performance, XpertFramework incorporates database caching mechanisms. This involves storing frequently accessed data in memory, reducing the need for repeated database queries. Redis is commonly employed as a caching layer in XpertFramework -based systems.

DATA MIGRATION AND VERSIONING:

XpertFramework database architecture accommodates seamless data migration when upgrading the system. It also supports versioning, allowing the tracking of changes to the database structure over time.

DATABASE MIGRATION AND UPDATES:

XpertFramework Bench Commands:

1. XpertFramework provides commands for migrating the database schema.
2. **`bench migrate`** for applying changes to the database structure.

BEST PRACTICES:

Data Validation:

Server-Side Validation:

1. Leverage Xpert Framework to implement server-side validation.
2. Ensure data consistency and adherence to business rules.

Regular Backups:

Backup Strategies:

1. Regularly backup the database to prevent data loss.
2. Utilize Frappe Bench commands for automated backup tasks.

Indexing for Performance:

Query Optimization:

1. Use appropriate indexes to optimize query performance.
2. Consider indexing fields frequently used in search and filtering.

CHAPTER 7

SYSTEM COMPONENTS

INTRODUCTION TO SYSTEM COMPONENTS:

The XpertFramework is composed of various interconnected components that work collaboratively to deliver its functionalities. Understanding these components is essential for system administrators to effectively manage, customize, and troubleshoot the system. In this chapter, we will provide a detailed overview of the key system components within the XpertFramework.

XpertFramework App:

4. An application or module within the XpertFramework.
5. Can be a core XpertFramework app or a custom app developed for specific business requirements.

Roles:

1. Defines business logic, user interfaces, and database structures.
2. Can be customized or extended to accommodate unique workflows.

XpertFramework Modules

1. Individual units of functionality within a XpertFramework app.
2. Examples include ASC, SMC, and GSP modules.

XpertFramework Modules

1. The user interface framework within XpertFramework.
2. Provides a responsive and modular environment for interacting with XpertFramework apps.
3. Hosts modules and allows users to create, view, and manage documents.
4. Supports customization for tailoring the user interface to specific needs.

BACKGROUND JOBS:

Queues

1. Queues for handling asynchronous tasks.
2. Examples include email notifications, data imports, and report generation.
3. Ensures efficient handling of time-consuming tasks without affecting user experience.
4. Supports prioritization and scheduling of background jobs.

FILE MANAGEMENT:

File Attachments

1. Allows users to attach files (documents, images, etc.) to records.
2. Utilizes the XpertFramework file manager for efficient storage and retrieval.
3. Facilitates documentation and collaboration by attaching relevant files to records.

WEB SOCKETS:

Real-Time Communication

1. WebSockets enable real-time bidirectional communication between the server and clients.
2. Used for instant updates, notifications, and collaboration.
3. Enhances user experience by providing live updates on data changes.
4. Supports features like live chat and collaborative editing.

INTEGRATIONS AND APIS:

RESTful APIs

1. RESTful APIs provide a standardized way for external systems to interact with XpertFramework.
2. Enable CRUD operations (Create, Read, Update, Delete) over HTTP.
3. Facilitate integration with third-party services, mobile applications, or external systems.
4. Support data exchange between XpertFramework and external platforms.

SYSTEM CONFIGURATION:

Site Configuration

1. Configuration settings specific to each XpertFramework site.
2. Includes database connection details, email settings, and system preferences.
3. Allows system administrators to customize and fine-tune the behavior of the system.
4. Essential for adapting the system to organizational requirements.

USER MANAGEMENT:

Creating and Managing Users:

Effective user management is a fundamental aspect of administering a XpertFramework-based system. In this chapter, we will explore the processes involved in creating and managing users.

User Creation:

1. Navigate to the Users module in the XpertFramework Desk.
2. Click on "New" and fill in the required user details, such as username, full name, email, and password.

USER ROLES AND PERMISSIONS:

Role Assignment:

1. Assign roles to users based on their responsibilities.
2. Roles define the permissions granted to users, determining what actions they can perform within the system.

Permission Manager:

1. Utilize the Permission Manager in the XpertFramework Desk to configure granular permissions for users and roles.
2. Define access permissions for specific modules, doctypes, and fields.

USER AUDITING:

User auditing is crucial for tracking user activities and maintaining accountability.

Audit Log:

1. Enabling Audit Log:

Activate the Audit Log feature in the site configuration to record user actions.

2. Viewing Audit Log:

Access the Audit Log module in the XpertFramework Desk to review user activities and system changes.

CHAPTER 8

CODING GUIDELINES

INTRODUCTION TO CODING GUIDELINES:

Efficient and standardized coding practices are crucial for the development and maintenance of the application. This chapter outlines coding guidelines tailored for the XpertFramework, providing system administrators and developers with a set of best practices to enhance code quality, readability, and maintainability.

PYTHON CODING GUIDELINES:

PEP 8 Compliance:

Coding Style:

1. Adhere to the Python Enhancement Proposal 8 (PEP 8) style guide.
2. Consistent indentation, naming conventions, and formatting.

Modularization:

Use of Functions and Classes:

1. Promote modular design by encapsulating functionality within functions and classes.
2. Enhances code readability and reusability.

Comments and Documentation:

Inline Comments:

1. Provide clear and concise comments for complex code sections.
2. Document function purpose, parameters, and return values.

DocStrings:

1. Include docstrings for classes, functions, and modules.
2. Facilitates automated documentation generation.

JAVASCRIPT AND HTML/CSS CODING GUIDELINES:

Consistent Coding Style

JavaScript

1. Follow a consistent coding style for JavaScript code.
2. Use meaningful variable and function names.

HTML and CSS:

1. Maintain consistency in HTML structure and CSS styles.
2. Use classes and IDs judiciously for styling and scripting

Client-Side Validation:

JavaScript for Data Validation:

1. Implement client-side validation using JavaScript where applicable.
2. Enhances user experience by providing instant feedback.

CHAPTER 9

BACKUP & RESTORE PROCEDURES

IMPORTANCE OF REGULAR BACKUPS:

Data is a critical asset for any Frappe Framework-based system, and ensuring its safety through regular backups is paramount. This chapter delves into the significance of backups and provides comprehensive procedures for both backup creation and restoration.

Data Loss Scenarios:

Hardware Failures:

Unexpected hardware issues leading to data corruption or loss.

Software Glitches:

Application bugs, errors, or updates causing unintended data changes.

Security Threats:

Protection against data breaches, ransomware, or malicious activities.

Backup Storage:

Local Storage:

Store backups on the local server for quick access.

Off-site Storage:

Implement off-site storage solutions (cloud storage, remote server) for redundancy and disaster recovery.

Encrypted Backups:

Secure backups with encryption to protect sensitive data.

RESTORE PROCEDURES:

In the event of data loss or system failures, having efficient restore procedures is crucial for minimizing downtime.

Bench Commands:

Restore from Backup:

Use the Frappe Bench command-line tool to restore from a specific backup.

bench --site <site-name> --force restore /path/to/backup-file.

Interactive Restore:

Perform an interactive restore, selecting specific components to restore.

bench --site <site-name> --force restore –interactive.

Point-in-Time Restore:

Database Rollback:

Rollback the database to a specific point in time to undo changes.

bench --site <site-name> restore --force --with-public-files /path/to/backup-file --reinstall

Transaction Logs:

Utilize transaction logs for point-in-time recovery for databases that support this feature.

CHAPTER 10

TROUBLESHOOTING & GUIDELINES

LOGGING AND MONITORING:

Enable Debugging:

1. Configure XpertFramework to log debug information for thorough issue diagnosis.
2. Adjust logging levels based on the severity of the problem.

Log Files:

1. Familiarize yourself with key log files, such as **`web.error.log`.**
2. Regularly review logs to identify anomalies and errors.

MONITORING TOOLS:

Use Monitoring Solutions:

1. Implement external monitoring tools to track system performance.
2. Monitor key metrics, such as CPU usage, memory, and response times.

DATABASE ISSUES:

Identify Slow Queries:

1. Utilize database profiling tools to identify slow-performing queries.
2. Optimize queries or add appropriate indexes to improve performance.

CONNECTION ISSUES:

Database Connection Pooling:

1. Adjust database connection pooling settings to handle varying loads.
2. Monitor and address connection pool exhaustion.

BACKGROUND JOBS:

Job Queue Monitoring:

1. Monitor the job queue for stuck or failed jobs.
2. Investigate and resolve issues causing job failures.

Prioritize Critical Jobs:

1. Prioritize background jobs based on their criticality.
2. Adjust job priorities to ensure essential tasks are executed promptly.

REAL-TIME COMMUNICATION (WEBSOCKETS):

WebSockets Connection Debugging:

1. Troubleshoot WebSocket connection issues using browser developer tools.
2. Verify WebSocket configuration and resolve connection problems.

INTEGRATION AND API ISSUES:

Error Handling:

1. Implement robust error handling mechanisms for API requests.
2. Provide meaningful error messages for easier diagnosis.

10.9 Authentication and Authorization:

10.9.1 API Key Management:

1. Ensure secure handling of API keys for external integrations.

2. Review and update authentication mechanisms regularly.