**System Architecture Plan**

**Wide Angle Insights**

**Version 2.0**

Table of Contents

[Introduction 3](#_Toc171067095)

[Purpose 3](#_Toc171067096)

[Scope 3](#_Toc171067097)

[Overview 3](#_Toc171067098)

[System Requirements 4](#_Toc171067099)

[Functional Requirements 4](#_Toc171067100)

[Non-Functional Requirements 6](#_Toc171067101)

[Regulatory Requirements 6](#_Toc171067102)

[High-Level Architecture 7](#_Toc171067103)

[Client-Server Architecture 7](#_Toc171067104)

[Communication Flow 8](#_Toc171067105)

[Detailed Architecture Components 9](#_Toc171067106)

[Database Layer 9](#_Toc171067107)

[Detailed Design and Documentation 11](#_Toc171067108)

[Data Flow Diagrams (DFD) 11](#_Toc171067109)

[Development and Testing Strategy 15](#_Toc171067110)

[Development Workflow 15](#_Toc171067111)

[Testing Strategy 17](#_Toc171067112)

[Signature Panel 17](#_Toc171067113)

# Introduction

## Purpose

The purpose of this document is to outline the system architecture for Wide-Angle-Insights v.2.0, web-based application developed to manage clinical studies in efficient manner. This plan includes the high-level design, detailed components, infrastructure, and best practices for security and scalability.

## Scope

This document covers the architecture for the entire system, including client-side, server-side, database, security, and deployment strategies.

## Overview

The Wide-Angle-Insights aims to provide seamless experience of all below mentioned components

# System Requirements

## Functional Requirements

* **User Management (UMT)**
* Create and Assign User Roles.
* Request for User Activation and Deactivation.
* Activate and Deactivate Users.
* Maintain audit trails of user accesses, activation, deactivation and change.
* **Design Bench**
* Design forms/ modules.
* Design listings.
* Design rules for data cleaning.
* Review and Freeze process.
* Maintain audit trails of create, update and delete actions.
* **Interactive Web Response System (IWRS)**
* Kit (Investigational Product (IP) supply) management from IP storage facility i.e depot to investigator sites and finally to the subject and aids in drug accountability.
* Subject screening/registration, randomization, kit / treatment allocation as required by the study protocol.
* Data Correction through Data Change Form (DCF) in controlled manner.
* Handle Unblinding in case emergency
* Maintain audit trails of all the activities performed in the system.
* **eSource/Direct Data Capture (DDC)**
* Subject registration – Single as well as Re-registration as required by protocol.
* Review (at form level and / or visit level) and sign the records electronically (electronic signature) as defined for the study.
* Data once entered in DDC (eSource) by investigator site user flows automatically to EDC except contextual notes and uploaded documents.
* Source data review by monitor.
* Query resolution by Site user
* **Data Management/Electronic Data Capture (EDC)**
* Data is cleaned in EDC system through query management and later transferred for statistical analysis.
* Relevant data from EDC flows into the pharmacovigilance / safety portal as per mapped fields and in “Medical Monitoring” as per defined listings.
* This aims to capture data only once (single source of truth) and improve efficiencies of all stakeholders in the clinical trial including investigator site teams, site monitors, data management personnel, medical monitors etc.
* **Pharmacovigilance (PV)**
* This system is a system to record, process and report Serious Adverse Events (SAE) in clinical trials.
* This system allows investigator to execute electronic signatures for SAEs.
* **Medical Monitoring (MM)**
* This system is a Real Time Integrated Medical Review platform that helps medical monitors to review safety and efficacy data in a clinical trial in a meaningful, organized, and comprehensive manner by bringing all the related information required for medical adjudication of a possible signal at one single place.
* **Electronic Trial Master File (eTMF)**
* This system is an internet based, configurable, application that acts as a repository of the essential documents of a study.
* This system follows the hierarchy of three-layered structure based on CDISC (Formerly known as DIA) TMF reference model structure of three layers i.e. Zone, Section and Artifacts.
* This system can store PDF as well as excel (tracking tools), data files, media, digitized content.
* This system allows user to perform QC review of the uploaded documents by verifying correct nomenclature, legibility of document/s, correct orientation, correct location, attributable, complete, original / certified.

## Non-Functional Requirements

* Physical and logical security should be in place to ensure that access to the computerized system and to the data is restricted to authorized personnel.
* Information stored in the system should not be altered, browsed, queried, or reported via external software applications that do not enter through the protective system software.
* Staff should be thoroughly aware of system security measures and the importance of limiting access to authorized personnel.
* Data should be backed up regularly in a way that would prevent a catastrophic loss of the data.
* Controls should be in place to prevent, detect, and mitigate effects of computer viruses on data and software.
* Both business and transactional data should be backed up regularly in a way that would prevent a catastrophic loss of the data.
* All business and transactional data generated and managed using the system must be encrypted using secure digital encryption keys to prevent data theft.
* The system infrastructure must be protected from all environmental hazards such as fire, water, electrical failures, etc.
* The organization must have tested procedural controls for managing system risks and incidents throughout the system lifecycle.
* The organization must have uninterrupted technical and functional support to handle all system issues reported by business users.
* The system must have disaster recovery capabilities to resume business operations in the event of a disaster.

## Regulatory Requirements

* The system must be compliant with 21 CFR Part 11 requirements.

# High-Level Architecture

## Client-Server Architecture

* **Client Side (Frontend)**

**Components -**

* Technologies: HTML, CSS, jQuery and JavaScript.
* Responsibilities: User interface and client-side validation.
* Web Browser/Mobile App: The interface through which users interact with the application.
* HTTP Requests: Communication initiated by the client to request data or services from the server.
* **Server Side (Backend)**

**Components -**

* Technologies: ASP.NET, C#.
* Responsibilities: Business logic and data processing.
* Data Access Layer (DAL): Interfaces with the database, executing queries and returning results.
* Database: Stores application data typically managed using SQL Server.

## Communication Flow

* **Client Request**
* The client (browser or mobile app) sends an HTTP request to the server.
* This could be a GET request to fetch data or a POST request to submit data.
* **Request Handling**
* The request has been received by an ASP.NET Controller.
* The Controller interacts with the Service layer to process the request.
* **Business Logic Execution**
* The Service layer contains the core business logic.
* It may perform operations such as validating input data, processing business rules, or orchestrating complex workflows.
* **Data Access**
* The Service layer interacts with the Data Access Layer (DAL) to retrieve or modify data in the database.
* The DAL uses Entity Framework to communicate with the SQL Server database.
* **Database Interaction**
* The database executes the SQL queries or commands and returns the results to the DAL.
* Results could be data rows for SELECT queries or status indicators for INSERT, UPDATE, or DELETE operations.
* **Response Handling**
* The DAL returns data or status back to the Service layer.
* The Service layer processes this data as needed and passes it to the Controller.
* **Client Response**
* The Controller prepares the final response, which could be an HTML page, JSON data, or other formats.
* The response is sent back to the client.

A diagram of a communication flowchart

Description automatically generated

# Detailed Architecture Components

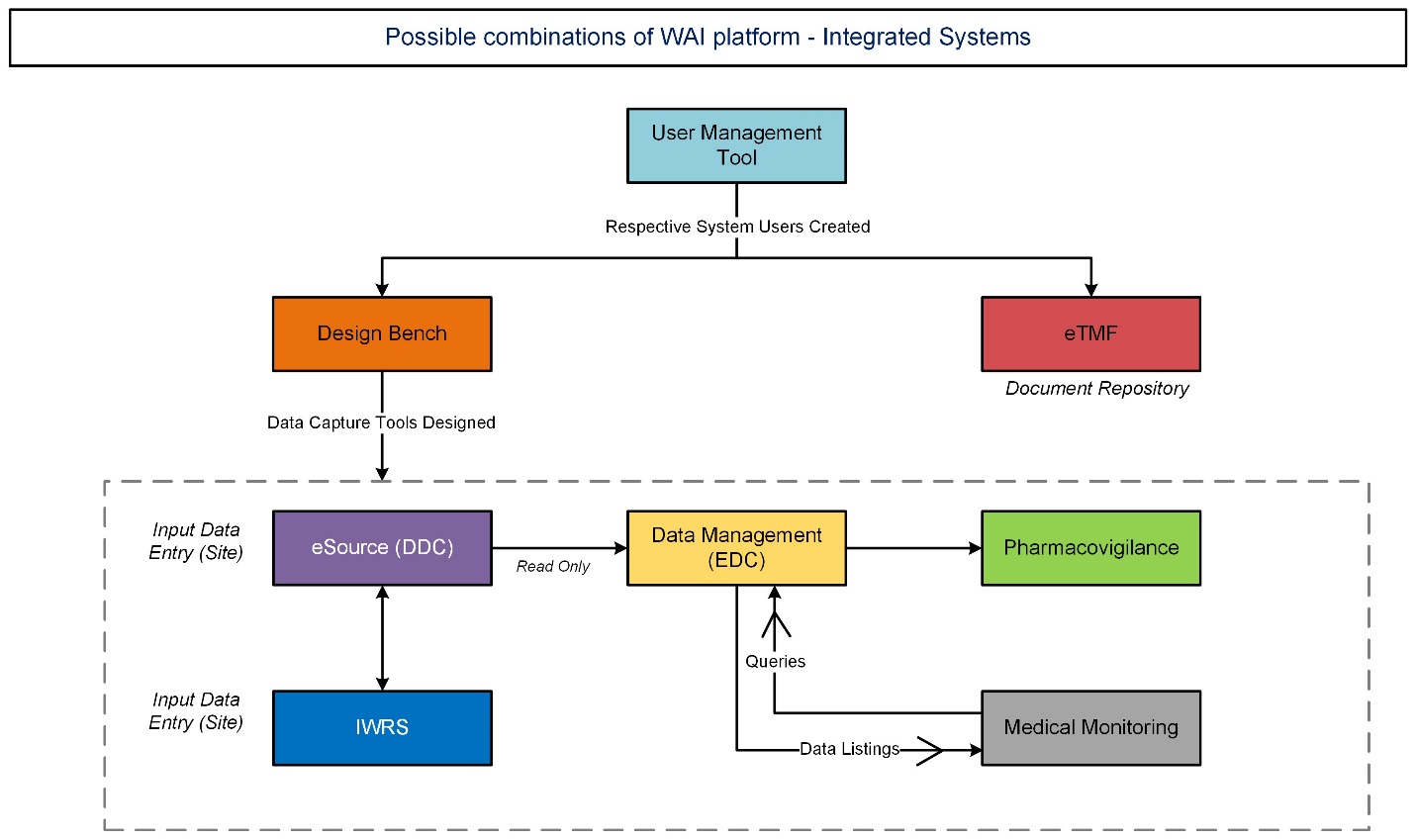
## Database Layer

* **Database Choice**: SQL Server.
* **Schema Design**
* The database schema is designed to support all the above mentioned Functional and Non-functional requirements.
* The schema ensures data normalization and integrity, with appropriate relationships and constraints.
* **Indexing Strategy**
* Primary key indexes on all primary key columns to ensure efficient retrieval.
* Foreign key indexes to speed up join operations between related tables.
* Additional indexes on columns frequently used in search and filtering operations.
* **Data Integrity and Constraints**
* NOT NULL constraints on all essential fields to ensure data completeness.
* UNIQUE constraints on fields that require unique values such as User ID.
* FOREIGN KEY constraints to enforce referential integrity between related data entities.
* CHECK constraints to ensure valid data values.
* **Data Access Layer (DAL)**
* Entity Framework Core: ORM for data access.
* DbContext: Central class to manage database connections and queries.
* **Backup and Recovery**
* Backup Strategy:
* Regular automated backups using SQL Database Backup service.
* Daily full backups, hourly differential backups, and transaction log backups every 10 minutes.
* Recovery Plan:
* Documented process for database restoration in case of failure.
* Regular testing of backup and restore procedures.
* **Data Security**
* Encryption:
* Use Transparent Data Encryption (TDE) for data at rest.
* Use SSL/TLS for data in transit.
* Access Control:
* Implement role-based access control (RBAC) to restrict access to sensitive data.
* Use Active Directory (AAD) for managing user roles and permissions.

# Detailed Design and Documentation

## Data Flow Diagrams (DFD)

**Integrated Instance**- User Management Tool (UMT), Design Bench (DB), eSource (DDC), Data Management (EDC), Pharmacovigilance (PV/Safety), Interactive Web Response System (IWRS), Medical Monitoring (MM) and electronic Trial Master File (eTMF)



**EDC Integrated Instance-** UMT, DB, EDC, PV/Safety, MM and eTMF.

A diagram of a system

Description automatically generated

**EDC Standalone**- UMT, DB, EDC, and MM.

A diagram of a software company

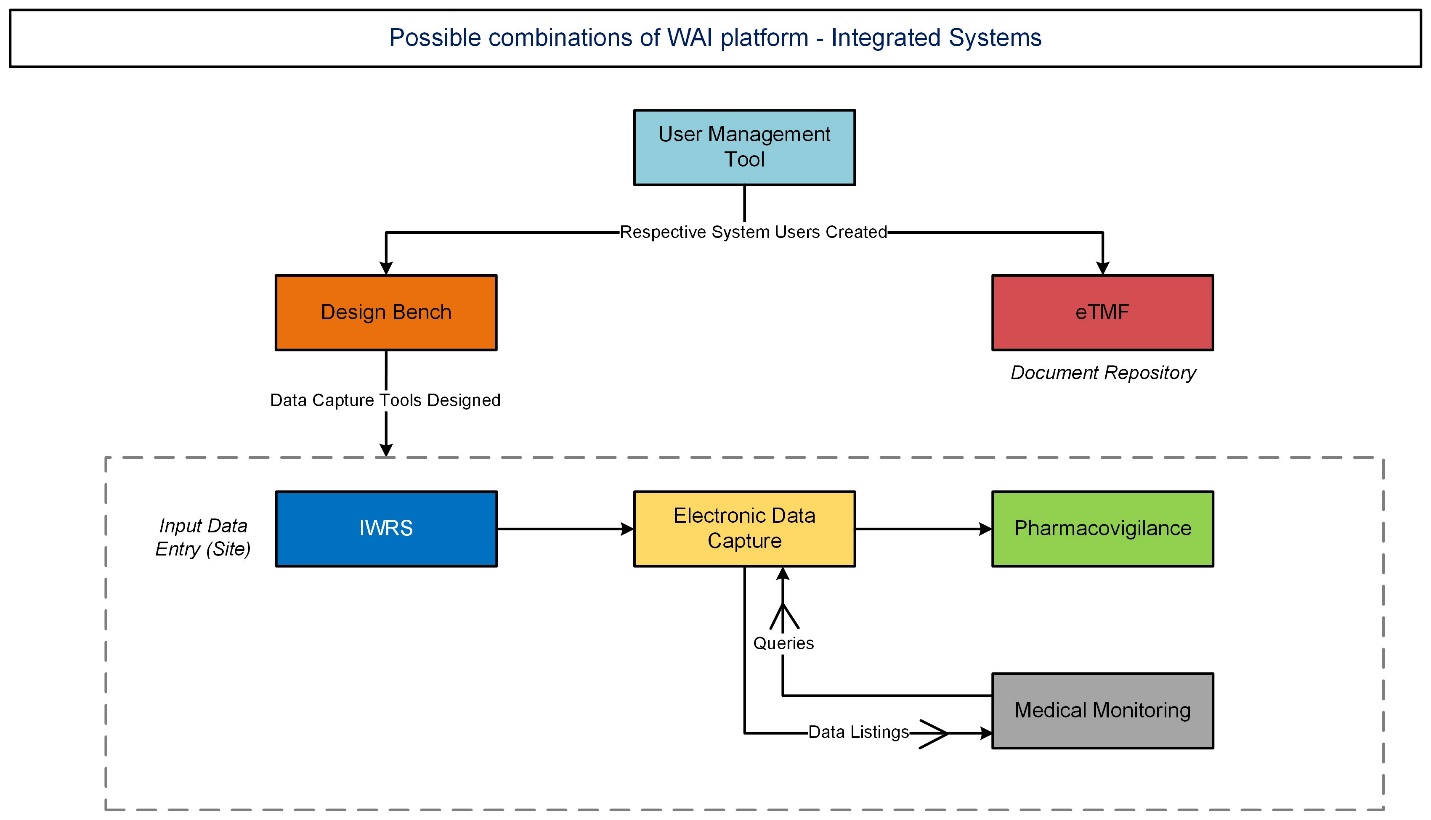
Description automatically generated with medium confidence

**eTMF Standalone**- UMT and eTMF

A close-up of a computer screen

Description automatically generated

IWRS Integrated- UMT, electronic Trial Master File (eTMF)



**IWRS integrated**-User Management Tool (UMT), Design Bench (DB), Data Management (EDC), Pharmacovigilance (PV/Safety), Interactive Web Response System (IWRS), Medical Monitoring (MM) and electronic Trial Master File (eTMF)

A diagram of a system

Description automatically generated

**IWRS Standalone** - User Management Tool (UMT), Design Bench (DB), Interactive Web Response System (IWRS)A diagram of a company

Description automatically generated

**PV Standalone** - User Management Tool (UMT), Design Bench (DB), Pharmacovigilance (PV/Safety)

A diagram of a diagram

Description automatically generated

# Development and Testing Strategy

## Development Workflow

* **Requirement Gathering and Analysis**
* Gather Requirements: Meet with stakeholders to understand the application requirements.
* Analyze Requirements: Create user stories, use cases, and functional requirements.
* **System Design**
* Architectural Design: Define the overall system architecture, including high-level components, databases, and interactions.
* Detailed Design: Create detailed designs for each component, including class diagrams, database schemas, and user interface mockups.
* **Environment Setup**
* Development Environment: Set up the development environment with necessary tools (e.g., Visual Studio, SQL Server Management Studio, DevOps).

Database Configuration: Configure the SQL Server database, including schema creation and initial data seeding.

* **Development**
* Feature Development: Implement features in iterations following the Agile methodology.
* *Sprint Planning*: Plan each sprint with specific goals and tasks.
* *Development*: Write code for backend (ASP.NET Core) and frontend (if applicable).
* *Code Review*: Conduct peer reviews to ensure code quality and consistency.
* Database Development: Implement database changes using Entity Framework or SQL scripts.
* Integration: Integrate frontend with backend services, and backend with the database.
* **Testing**
* Unit Testing: Write and run unit tests for individual components to ensure they function as expected.
* Integration Testing: Test the integration between different components.
* System Testing: Perform end-to-end testing of the entire application to ensure all components work together seamlessly.
* User Acceptance Testing (UAT): Conduct testing sessions with stakeholders to validate that the application meets their requirements.
* **Deployment**
* Staging Environment: Deploy the application to a staging environment that mimics the production environment.
* Testing in Staging: Conduct final testing in the staging environment.
* Production Deployment: Deploy the application to the production environment.
* **Maintenance**
* Monitoring: Continuously monitor the application for performance and errors.
* Bug Fixes: Address any bugs or issues that arise in production.
* Updates: Implement new features and improvements based on user feedback.

## Testing Strategy

* **Unit Testing**
* Scope: Test individual units/components.
* Tools: Use tools like Azure DevOps to manage and track test cases.
* Approach: Write tests to cover various scenarios.
* **Integration Testing**
* Scope: Test the interaction between integrated components.
* Tools: Use tools like Azure DevOps to manage and track test cases.
* Approach: Write tests that validate data flow and interactions between components.
* **System Testing**
* Scope: Perform end-to-end testing of the application.
* Tools: Use tools like Azure DevOps to manage and track test cases.
* Approach: Create test cases that cover all user stories and workflows.

## Signature Panel

|  |  |  |
| --- | --- | --- |
|  | **Name** | **Signature and Date** |
| **Author** |  |  |
| **Reviewer and Approver** |  |  |