



Rule of Law Project (ROLP) - Judges Affairs Unit Application (JAUA)

System Software Architecture

Version [1.1]

1 REVISION SHEET

Change & Review History

Date	Version	Comment
22-Feb-2011	1.0	Document Created
5-Mar-2011	1.1	Modifications and Feedback applied: <ul style="list-style-type: none">- Security: authentication can be configured to be against active directory and database users.

TABLE OF CONTENTS

1	Introduction	4
2	Solution Architecture	5
2.1	Conceptual View	5
2.1.1	Application Users	6
2.1.2	Application Layers	6
2.2	Logical View	6
2.2.1	JUA Components.....	7
2.3	Technology View	13
2.3.1	JUA Technologies.....	13
2.4	Integration Approach.....	14
2.5	Reporting Mechanism.....	15
2.5.1	Technology.....	15
2.5.2	Report Generation Mechanism.....	16
2.6	System Performance	17
2.7	Rules and Constraints	17

2 INTRODUCTION

Application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.

This document is intended to provide an overall description of the JAUJ solution Software Architecture; it provides detailed description of the architecture particulars and technology specifications through the following views and topics:

- Conceptual View.
- Logical View.
- Technology View.
- Integration Approach.
- Reporting Mechanism.
- System Performance.
- Rules and Constraints.

3 SOLUTION ARCHITECTURE

The system architecture particulars and technology specifications will be detailed in the following sections.

3.1 CONCEPTUAL VIEW

Below is the conceptual view of the overall JAUA solution, the conceptual model defines the modules and supporting capabilities needed to realize the functional and non-functional objectives of the solution.

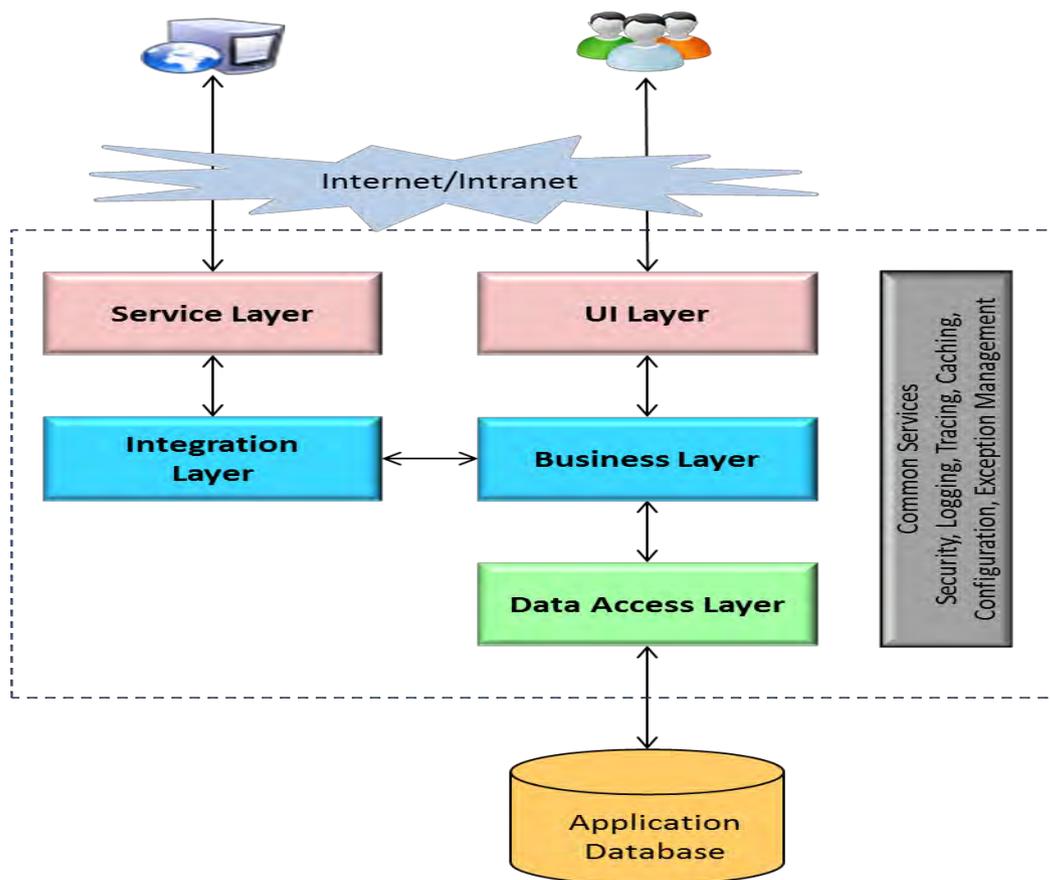


Figure 1: Conceptual Model

The conceptual model is explained in the following sub-sections.

3.1.1 APPLICATION USERS

These are the actors who will interact with and benefit from the JAUA solution. Those include:

- **Business Users:** Will be accessing the solution web interface through the Internet/Intranet to perform their daily activities that relate to their business.
- **External Systems:** Will be accessing the solution integration services through the Internet/Intranet to integrate with JAUA system.

3.1.2 APPLICATION LAYERS

These Services represent the collection of services that are implemented through the system, which includes the core functionality of the system, the services are:

- **UI Layer:** provides the visual interface frontend to be used by the various personnel at the organizations to access and interact with JAUA solution.
- **Business Layer:** responsible about performing all business operations.
- **Data Access Layer:** responsible for data persistence and query and communicating with other systems.
- **Service Layer:** responsible about exposing the required system data and functions to integrate with other systems.
- **Integration Layer:** responsible about structuring and implementing the data and functions that should be exposed to integrate with other systems.

3.2 LOGICAL VIEW

The logical view depicts how the solution meets the requirements in providing a set of business services. Our solution adopts an n-tier architecture model categorizing its main functions into separate modules. This provides the overall with high abilities for extensibility, interoperability, and high maintainability.

The system realizes an open architecture utilizing the latest industry standards and technologies. The following diagram shows a high level logical view of the main components comprising the JAUA solution.

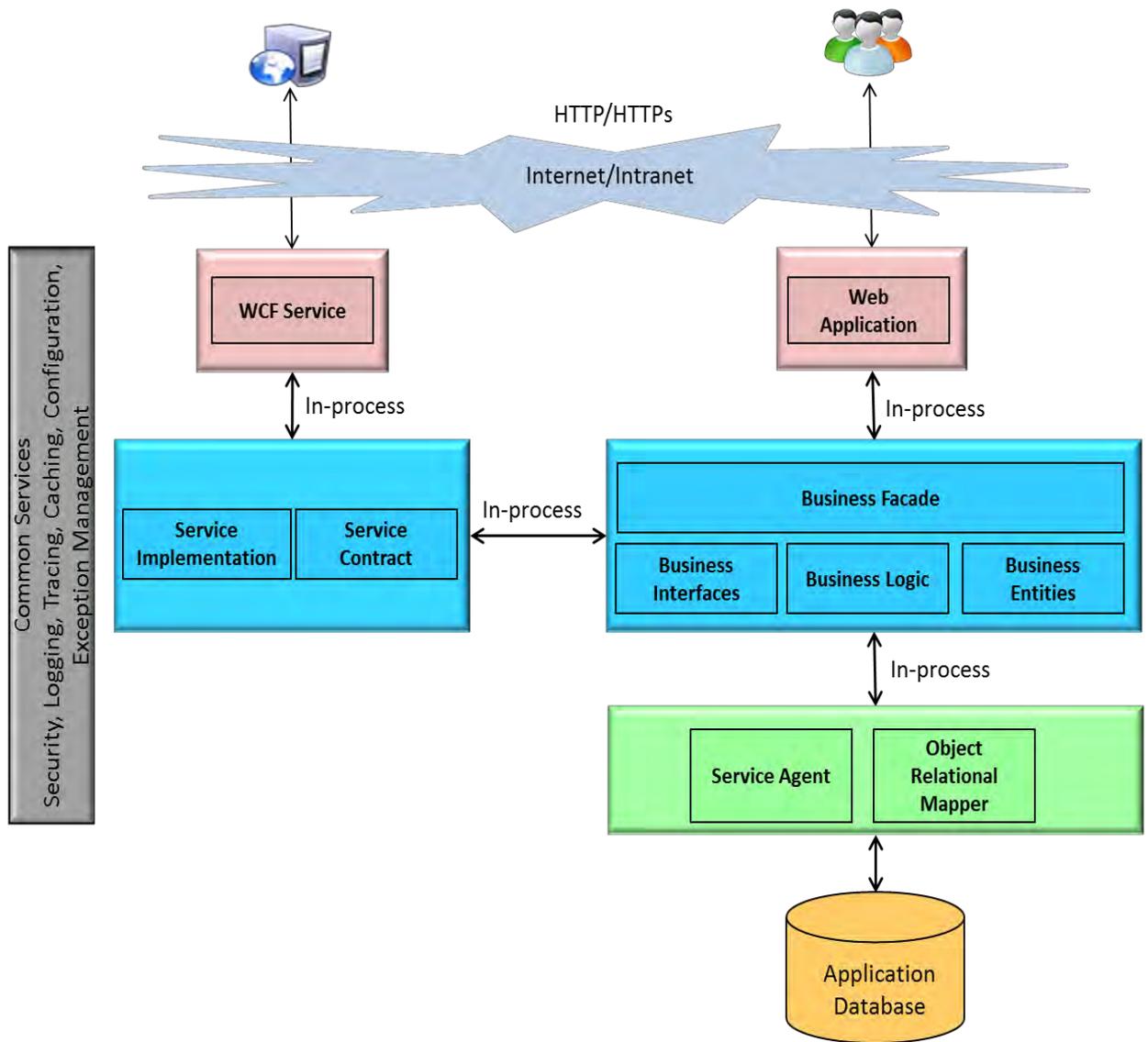


Figure 2: Solution Architecture

In the following subsections these components are explained in detail.

3.2.1 JAUA COMPONENTS

This section elaborates on JAUA layers components:

3.2.1.1 UI LAYER COMPONENTS

The UI layer contains the web application component which provides the end user access to the system, validate user inputs, and to communicate with the business components to process user actions and inquires.

The web application architecture is hardened by Asynchronous communication components to achieve high performance and the best user experience. Additionally, reusable UI controls are being used to increase the productivity and speed up the implementation. The web application architecture defines the following components:

- **Asynchronous Communication:** These components will be used to enhance the end-user experience, they are necessary to support partial web page rendering and back-end server requests to provide the user with seamless interaction with the web page. This asynchronous communication layer is the foundation for a client Web application presentation that can be significantly more responsive than the traditional one.
- **User Interface Views:** These are the pages and the user interface elements (controls) that capture and validate user input, manage interactions, and display information. They also contain site map, menus, layout and design elements.

3.2.1.2 BUSINESS COMPONENTS

The business components are responsible to process the business functionalities and validate the user inputs based on business processes, rules and constraints.

The business components layer contains the business functionality. These components are described in the following points:

- **Business Façade:** represents the entry point to business components. It is responsible about structuring the business processes and exposing business processes interfaces regardless of the details of the business process. In addition to that it simplifies the management of transactions and the security of business layer and allows adding other supporting services like tracing with minimal configuration.
- **Business Entities:** represents the system domain entities.
- **Business Logic:** represents the system functionalities and responsible to process them based on the business processes, rules and constraints.
- **Business Interfaces:** supports the decoupling between UI layer and Business Layer.

3.2.1.3 DATA ACCESS COMPONENTS

The Data Access Components are responsible for data persistence and query and communicating with external and legacy systems. It contains the following modules:

- **Object Relational Mapper:** It converts data between the incompatible database structures in the persistence tier and the object oriented application entities in the business tier. These components will be used for working and integrating with the application database.

- **Service Agents:** Service Agents are components that manage the semantics of communicating with external services that JAUA solution will integrate with.

3.2.1.4 SERVICE LAYER COMPONENTS

The service layer contains a WCF service to expose the required system data and functions to external systems which simplify the integration between them.

3.2.1.5 INTEGRATION LAYER COMPONENTS

The integration layer contains two components that are responsible about structuring and implementing the data and the functions that should be exposed to other systems:

- **Service Contract:** represents the functions and the entities that the system should expose to integrate with other systems.
- **Service Implementation:** represents the implementation of service contracts functions.

3.2.1.6 APPLICATION DATASTORE

The application data store is implemented through Oracle 10g or 11g database, and contains the Tables, Triggers, Views, Stored Procedures, Indexes, and Functions needed by the solution.

3.2.1.7 COMMON SERVICES

Contains business components shared by all other modules, functionality provided includes: Security, Configuration, Logging, Caching, Tracing, and Exception Management. A brief description about selected components is below.

3.2.1.7.1 SECURITY

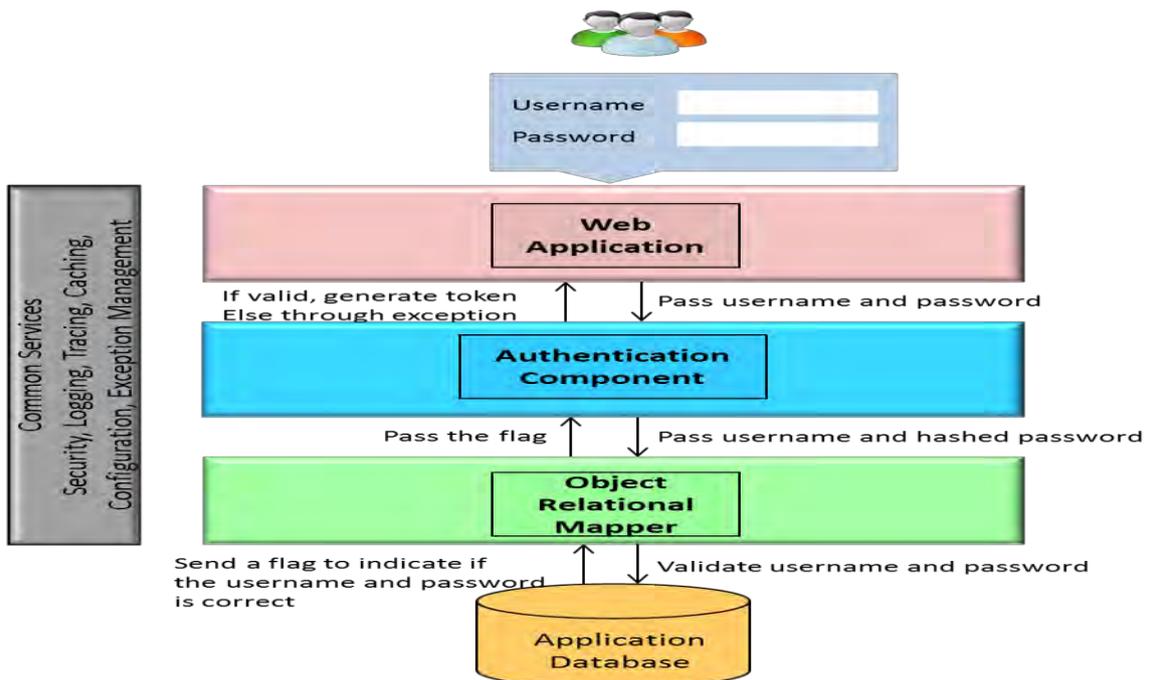
3.2.1.8 AUTHENTICATION

Whenever a user tries to access any area of the system, he is requested to go through an authentication process. The authentication process consists basically of a login process, in which the end user is required to explicitly submit his/her credentials, namely username and password or if the system is configured to use windows authentication it will automatically take the logged in windows user credential . The credentials are validated against the database of users who are allowed to access the system (these users can be populated directly from active directory users list). If the credentials are found to be correct and valid, the user is given access to the system; a token that contains details related to the user is generated and kept in the user's session on the server to flag that he is granted access to the system.

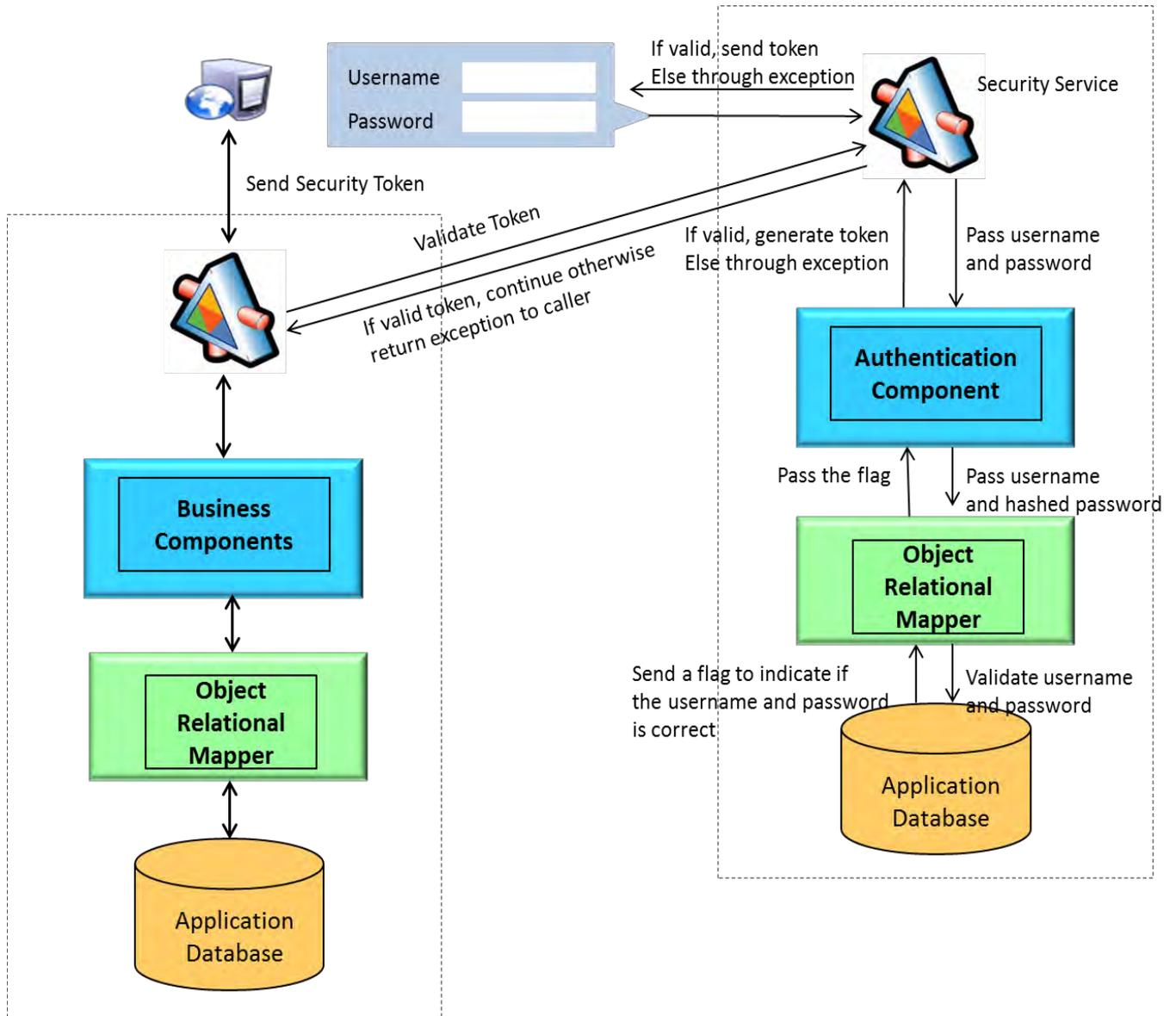
Whenever the user tries to access any resource of the system, the system looks for the user’s token, and if found, the user is allowed to access the resource. Else, the user will be required to go through the authentication process.

The user’s token will be available as long as the user’s current session is available and is active. If the session ends for some reason (by closing the browser window, for example), or if the session is idle for a long time (the default time is set to 20 minutes), the token will be removed and the user will be required to submit his/her credentials again to be able to use the system.

The authentication component built on top of .Net Framework security components to manage the authentication process and it supports both Forms and Windows authentication.



In case of there is a system needs to integrate with JAUA system, the authentication process will use custom security service to generate the security token which will be returned as WCF message.



3.2.1.9 AUTHORIZATION

Once the user is authenticated, he goes through an authorization process to determine the resources that he has access to.

The authorization process starts by determining what roles the user has. And then, determining what modules can be accessed by these roles. The system will display the modules that can be accessed by the user, and block access to any other module.

Within the module, the main menu will display only the system functions that the user can access based on his roles. Also, whenever any page is loaded, the system checks if this page belongs to a system function that the user is granted access to. If not, the system denies access to the page and prevents it from loading.

On the page level, the system checks what type of actions the user can perform in the page, and controls the page rendering accordingly.

3.2.1.10 DATA PROTECTION

The data protection is applied on both the communication channel and the data itself.

The communication channel can be made (optionally) secured by using the SSL to protect the communication between the system and the end user/system to prevent data sniffing.

The data is protected by encrypting sensitive data in the database like users' passwords.

3.2.1.11 COMMON ATTACKS

The system applies the security best practices to prevent common security breaches and attacks like SQL Injection and Cross Site Scripting.

3.2.1.12 AUDITING

The system uses custom auditing component to track users' changes on the domain entities and to record their actions on the system. The audit details who did what, under which module, when and what was changed in terms of old value and new value.

3.2.1.12.1 LOGGING

Logging plays a crucial role in monitoring the system health and making sure all operations are running properly. In case an error occurs, log files help in tracing the function calls and parameters that may have caused the error.

The logging engine logs all exceptions that occur in the system, with their details and call stacks. The logging engine also logs informative details about large, heavyweight operations, such as reports

generation. Also, the logging engine keeps track of all database calls when needed, along with the parameters being passed, and the time needed for those calls to complete which helps in tracking database issues.

To ensure the logging engine will not add performance overhead to the system, all log files are kept small (~1 MB); once the file reaches its maximum size, it is automatically archived, and a new log file is created. This way, IO operations on log files are optimized for best performance.

3.2.1.12.2 EXCEPTION HANDLING

Exceptions are bound to happen in all systems. The point is to make sure exceptions are properly handled, and will not cause the system to fail.

Whenever an exception occurs, the exception details will be logged and a friendly error message appears to the user informing him that an error has occurred, and guides him on what he should do next.

Meanwhile, and depending on the hosting environment configurations, the system will send an email to the support team with all the technical details of the exception so that they can respond immediately to the exception and fix it.

3.3 TECHNOLOGY VIEW

In this section the solution is presented in terms of the technologies being used.

3.3.1 JAUA TECHNOLOGIES

The solution utilizes mainly the following technologies:

- Microsoft® .NET Framework 3.5 (Including ASP.NET, AJAX, Windows Services, and WCF)
- Microsoft® Internet Information Services 7.0
- ORACLE® 10g or 11 Database
- ASPOSE Reports
- HTML 4.0, CSS 2, JavaScript

Combining these products provides a robust, high performing and scalable solution. The following details describe how these products fit in the individual layers of the architecture.

- Users will use standard web browsers to use services. Browsers will communicate with the web server through the Internet over HTTP and optionally over HTTPS.
- The web application is built using ASP.NET, HTML, JavaScript, CSS, and AJAX. ASP .NET provides a feature-rich and high performance platform for building dynamic web applications. Microsoft AJAX extends ASP .NET capabilities to increase performance and usability of websites and deliver the desktop experience to the web interface user. CSS will be used to design the look and feel of the portal separating content from layout and formatting.
- The business components are implemented mainly using custom .NET code in addition to WCF (part of the .NET Framework) and hosted on IIS to implement the Integration web services.

The communication between the web application, the business components, and the persistence services will be in-process; meaning that the three layers should run on the same machine providing high performance while maintaining modularity and separation of concern.

- The data access components use ADO .NET (part of the .NET Framework class library) for data access and persistence in the application database. ADO .NET features high performance streaming methods for data access that do not require data transformation from one format to another. Service agents use custom proxy code to invoke the integration services exposed by other applications.
- The application data store is an Oracle 10g or 11g database.

3.4 INTEGRATION APPROACH

As can be depicted from the figure below, JAUA is integration ready; it provides the needed structure for two ways integration:

- Application business services can be invoked by consuming web services (implemented as WCF services) through a secure channel using a well-defined message contracts.
- JAUA can invoke other application's services and functions through its "Service Agents".

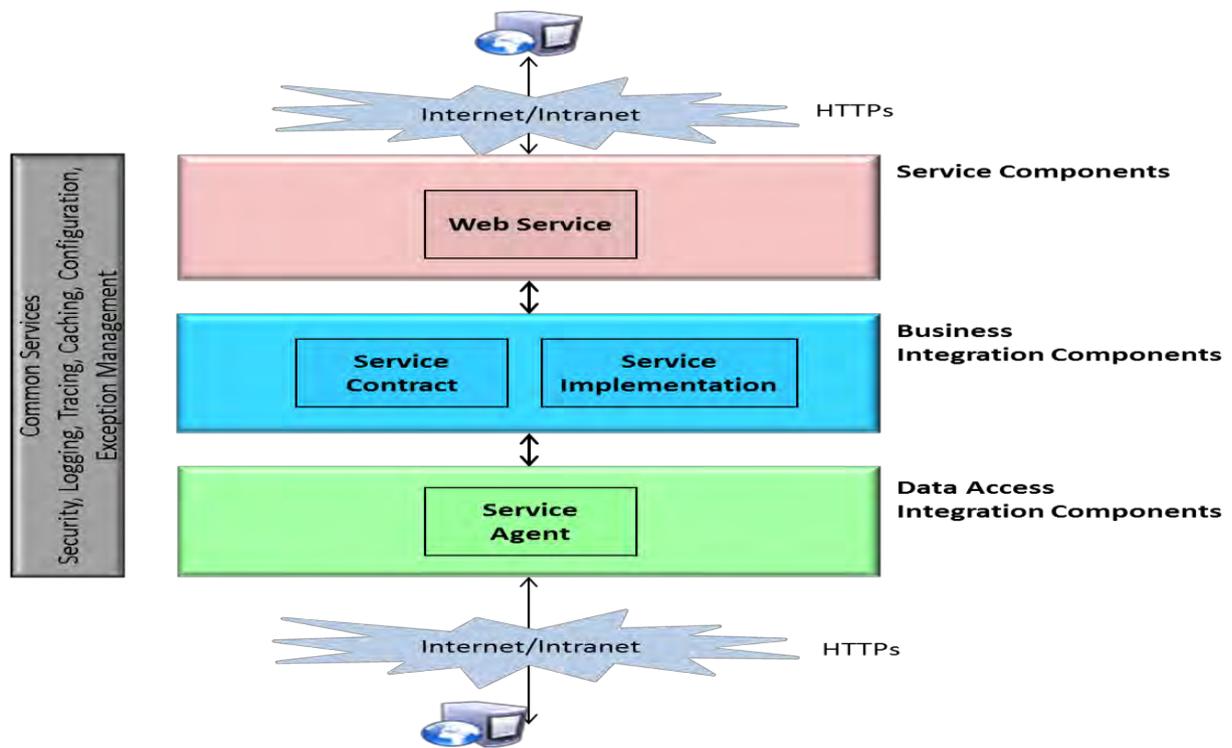


Figure 3: Solution Integration

3.5 REPORTING MECHANISM

JUAU offers a set of reports in different formats. The reports are designed to help the users have better understanding of their data. Reports are created in the background without affecting the user’s operation on the system.

3.5.1 TECHNOLOGY

3.5.1.1 ASPOSE REPORTING ENGINE

JUAU uses ASPOSE reporting engine to build and generate system reports. ASPOSE supports generating reports in multiple formats, including PDF, Excel and Word.

3.5.1.2 ASYNCHRONOUS GENERATION

In every system, reports generation is one of the most time-and-resource-consuming tasks. If there is a huge amount of data, a report may take several minutes to be generated.

Therefore, to avoid affecting the system’s performance, and in order not to interrupt the user’s work, the reports are generated by a Windows service running in the background. This service

monitors any reports requests and executes them immediately; while the user continues his/her work on the system. Furthermore, for better performance, this Windows service does not have to reside on the same server that is hosting the system; it can operate on its own server with its own resources.

3.5.2 REPORT GENERATION MECHANISM

Whenever a user requests a specific report in the system, a new record is created in the database containing the report type ID and user ID, along with all the parameters that the user specified, and a flag indicating that the report is **Pending** generation.

It is worth noting here that the reports are generated per user, which means that only the user who requested the report will have access to it; other users will not even know that the report is generated.

A Windows service running in the background keeps scanning the database for any reports that are flagged as pending generation. When it finds a report, it changes the flag to **In Progress**, and then it loads the information related to the requested report type.

Each report type has a code class that maintains the report's details, including its layout, database calls, etc. Based on the report type, the service determines which code class it should execute to generate the report.

Once the execution of the procedures in the report's class completes, the report file is saved to the hard drive, and the service flags the report's record in the database as **Complete**.

On the system's user interface, the reports page that contains a list of the reports that the user requested keeps updating its status automatically every 5 seconds (The refresh period is configurable). So, whenever a report flag is updated, it will be reflected immediately on the UI for the user to see.

If, for any reason, an error occurs during report generation, the service flags the report as **Completed with Errors**, and tries to regenerate it again during its next cycle on the database.

3.6 SYSTEM PERFORMANCE

In order to achieve the best performance out of the system, the system uses the common approaches to increase the system performance like caching non-transactional data, using asynchronous communication, using singleton pattern for creating stateless objects.

3.7 RULES AND CONSTRAINTS

1. The application should use a custom developed framework components and libraries. this framework contains many components that simplify the development process and save development efforts.
2. The application should validate the user inputs on the server side in addition to the basic client side validation.
3. All textual inputs should be HTML encoded to prevent XSS.
4. UI layer shouldn't contain any business logic code.