Instruction

This document provides the final detailed technical (infrastructure) specifications, technologies and design to address the business needs outlined in the *Business Requirements Analysis of the* JewelsFeedTracker application. The *High-Level Technical Design and Development* that was completed during the this phase of the JewelsFeedTracker should be used as a starting point for this more detailed template.

***If there are significant changes to requirements or design, this document should be updated accordingly and funding to support the changes should be verified and approved.***

1. Project Overview

Provide high-level information to track the various feed (was developed in PHP framework) into a centralized application in .Net core framework. End users understand the solution being proposed as well as the business reasons supporting the solution.

1.1 Objective

JewelsFeedTracker is a platform to track the various stones feed raw data received from their own apis and then provides a well-defined format after going through into many business rules and restrictions. Raw data and resulted data is saved into csv/exl format in application’s directory.

1.2 Business Case

The main objective of JewelsFeedTracker to develop a system where we can process the raw data that we are receiving from a Weservice, Api or csv/xl url and returning a well-defined data to store in our sql database. Old system was developed in curl/php had a lot of limitations like lacking of centralized procedure to formatting the data received from any api/websevice etc.

In our application, we are using .net core framework with Serilog(**logging framework for . NET**) and Hangfire Job Scheduer (**an open source job scheduling framework, to schedule fire-and-forget, recurring tasks in Web applications**) to develop an advance and centralized application for tracking the feed files on a regular interval that is managed by Hangfire framework.

1.3 Risks

Consuming the api/SOAP based webservice may lack to provide the raw data due to wrong url or expired/change the authentication credentials and resulting the failure of our data execution totally so we need to make sure the correct url/credentials before processing the feed url into job pipeline.

1.4 Feed Files Planning to process

The list of following files which are processed to store into the SQL database after manipulation.:

1. finestar.php
2. glowstar.php
3. dfe.php
4. hvk.php
5. harikrishna.php
6. rapnet.php
7. akarsh.php
8. dharmanandan.php
9. diamond\_srd.php
10. jbbrother.php
11. redexim.php
12. sagar.php

1.5. Technologies/Framework Used

Visual Studio 2019, .Net Core Framework -Version 3.1, C# - Version 7.1, Hangfire -Version 1.7.25,

Serilog.AspNetCore -Version 4.1.0, SQL 2008 R2 etc

2. Application Coding Structure Overview

In JewelsFeedTracker Application the proposed architecture is n-layer with factory design pattern with other web design concepts & advance utilities to develop a robust, modular and cross platform application. Below is some overview of various library project developed in c#, .Net core advance concepts.

1. JewelsFeedTracker.Web – This is the Web layer of our application which contains the .net core startup methods which initializes the feed files execution and integration of other layers. Hangfire and serilog packages are a part of this layer.
2. JewelsFeedTracker.Utility – We are using this contain few application’s utilities like webservice references that are being used in some web services to fetch the data.
3. JewelsFeedTracker.Security – We can declare the security measures and other confidential concepts like token, credentials in the application.
4. JewelsFeedTracker.Queries – This layer contains the factory methods to integrate the business rules on the feed files which further process them after the manipulation and then call repository for bulk saving operation(BCP)
5. JewelsFeedTracker.Data.Models – Models of feed files are declared in this layer.
6. JewelsFeedTracker.Data.Access – We are using this layer to use the common methods of database operations on the data and before sending to final data table to BCP transaction.
7. JewelsFeedTracker.Api.Common – Common array collection used by feed files is exits in this layer and we can also include common functions in this layer.

3. Technical Design Diagram

Diagram

Description automatically generated

**4. Technical Specifications**

4.1 Servers

4.1.1 Web Servers/Application Servers

An ASP.NET Core app runs with an in-process HTTP server implementation. The server implementation listens for HTTP requests and surfaces them to the app as a set of request features composed into an HttpContext.

Kestrel server is the default, cross-platform HTTP server implementation. Kestrel provides the best performance and memory utilization, but it doesn't have some of the advanced features in HTTP.sys.

IIS HTTP Server is an in-process server for IIS.

4.1.2 Database Servers

We are using SQL Database to store the processed feed data after applying the business

rules on the raw data.

**4.2 Databases**

List a brief description and function for each database that is needed, and include the following information:

* SQL Server 2008 R2 and Other version.
* Microsoft SQL Server Management Studio - 10.50.4000.0
* Memory requirements – 1 tb
* CPU requirements – 16 GB RAM
* Amount of storage needed the first 12 months (100 GB)

**5. Database Diagram:**

Below is the database Diagram of our DiamondsFactoryMgmt which depicts the structure of various

tables that are used to store the stone information and others to apply the business rules on feed

data.

****