Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Summer 2023

ERS
Equipment Reliability Stratagies

Nicholas Nguyen
Pablo Sanchez
Hoang Long Quan Nguyen
Viraj Vipinbhai Sabhaya

Revision History

Revision	Date	Author(s)	Description
0.1	6.21.2023	NN, PS, HN, VS	document creation

CONTENTS

List of Figures

List of Tables

1 Introduction

Equipment breakdown can cause delays in the production system. A vital machine breakdown may also cause the whole production line or the whole factory to be shut down completely, which can cost manufacturers hundreds of thousands of dollars per day. Moreover, it also can cause health, safety, and environmental accidents as well. According to research, around ninety percent of machine breakdown cases can be prevented with a good preventive maintenance plan. This is why Equipment Reliability Strategies Application was built for. It will help organizations, that own or operate process facilities, to manage all of their equipment in a professional and the most efficient way, to keep their equipment operating efficiently, minimize health, safety, and environmental accidents, and to avoid unexpected costly repairs in the future. The Equipment Reliability Strategies is made available commercially, targeting organizations that own or operate process facilities across many industries. People in these organizations, such as man- agers, engineers, and inspection and maintenance staff are direct users of this application. This application is a user-friendly web-based application, that uses the latest technologies for both front-end and back-end. To use this application users must create an account, and log in with authentication and access control. Based on their roles and granted permissions, the users will be able to create equipment, add or edit equipment properties, and calculate risks for each individual piece of equipment. From there, they will be able to create, update or delete, and manage default strategies, best practices, scenarios, actions, and mitigation, and create maintenance plans, inspection plans, and reports

2 System Overview

The Equipment Reliability Strategies Application consists of four layers, from top to bottom, which are the Web, Core, Model, and Database layers. These architectural "layers" are the top-level logical view, or an abstraction, of the design. Those four layers have related functionality associated with each other as shown in Figure 1 below.

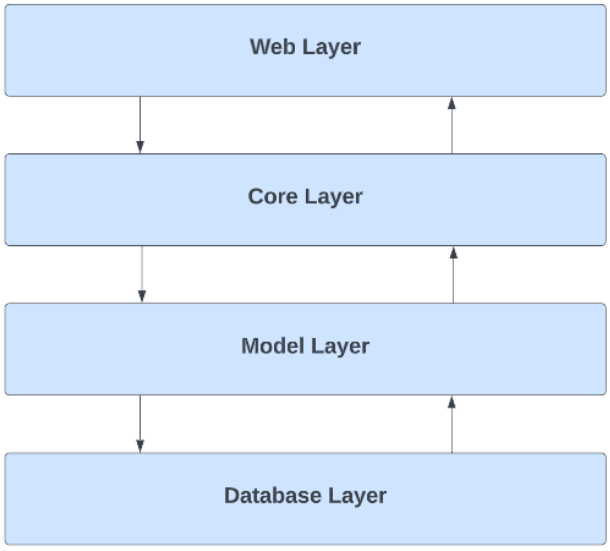

Figure 1: System architecture

3 Web Layer Subsystems

The Web Layer has the views, which is being presented to the User, helpers, language resources, security configurations, and the interact able parts of the web application. This layer will also mainly interact with the Core layer because it needs to know about the business objects which the Core layer can grab and give to the Web layer and has to be able to query the database which can be done with the Core layer as well.

3.1 Layer Operating System

The Web Layer should be compatible with most operating systems.

3.2 Layer Software Dependencies

The Web layer depends on the Model layer to access the business objects. The web layer depends on the Core layer to query the database.

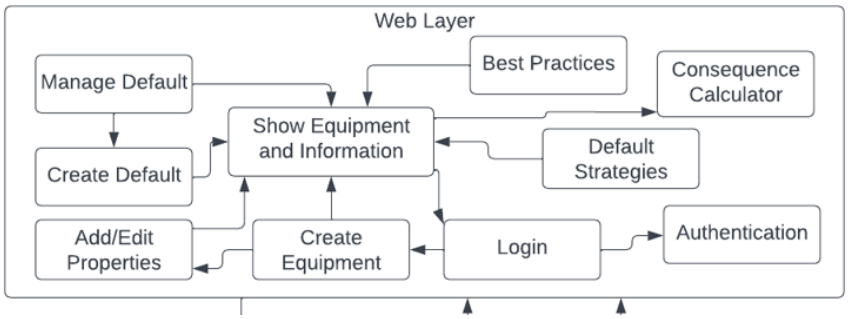

Figure 2: Example subsystem description diagram

3.3 Best Practice

The best practices subsystem will obtain a list of best practices to be used on a specific type of equipment.

3.3.1 Subsystem Operating System

Should be compatible with most operating systems

3.3.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.3.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.3.4 Subsystem Data Structures

Best Practice is a strategy that is put into the system and stored in a database table.

3.3.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.4 Manage Default

The Manage default subsystem allows the user to update an equipment's default whenever there is a change to it. It will also use the Core layer's Services subsystem to obtain the defaults so that they can be updated.

3.4.1 Subsystem Operating System

Should be compatible with most operating systems

3.4.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.4.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.4.4 Subsystem Data Structures

Defaults will be stored in a database table.

3.4.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.5 Create Default

The Create default subsystem will allow for users to create defaults for certain types of equipment to be shown to the user. It will interact with the Core layer's services subsystem to view the equipment's properties to know its type and to then save the defaults into the database tied with a certain equipment type.

3.5.1 Subsystem Operating System

Should be compatible with most operating systems

3.5.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.5.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.5.4 Subsystem Data Structures

Defaults will be stored in a database table.

3.5.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.6 Default Strategies

The Default strategies Subsystem will obtain a list of default strategies to be used on a specific type of equipment.

3.6.1 Subsystem Operating System

Should be compatible with most operating systems

3.6.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.6.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.6.4 Subsystem Data Structures

Strategy will be stored in a database table.

3.6.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.7 Create Equipment

The Create equipment subsystem allows users with proper access to create equipment that is being used within their units. It will use the Core layer for the authentication to make sure that the current user is allowed to create equipment for their unit.

3.7.1 Subsystem Operating System

Should be compatible with most operating systems

3.7.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.7.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.7.4 Subsystem Data Structures

Equipment will be stored in a database table.

3.7.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.8 Add/Edit Properties

The Add/Edit properties subsystem lets users update or add an equipmentâs properties so it can be displayed to everyone within the equipmentâs unit. It interact with the Service subsystem of the Core layer to be able to get the current properties of the equipment and either update them if they exist or add the new ones that were created.

3.8.1 Subsystem Operating System

Should be compatible with most operating systems

3.8.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.8.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.8.4 Subsystem Data Structures

Properties will be stored in the Equipment database table.

3.8.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.9 Login

The Login subsystem is used to log the user back into their specific account by using a previously created account and using the Core layer authentication subsystem to check if they used the correct login information.

3.9.1 Subsystem Operating System

Should be compatible with most operating systems

3.9.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.9.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.9.4 Subsystem Data Structures

User information will be stored in a database located in an external table.

3.9.5 Subsystem Data Processing

We have not implemented any specific algorithms for this subsystem

3.10 Authentication

Descibe at a high level the purpose and basic design of this subsystem. Is it a piece of hardware, a class, a web service, or something else? Note that each of the subsystem items below are meant to be specific to that subystem and not a repeat of anything discussed above for the overall layer.

3.10.1 Subsystem Operating System

Should be compatible with most operating systems

3.10.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.10.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.10.4 Subsystem Data Structures

Becht will be integrating the Login and Authentication for the web application

3.10.5 Subsystem Data Processing

We have not implemented any specific algorithms for this subsystem

3.11 Show Equipment and Information

The Show equipment and information subsystem displays the equipment and its corresponding information to the user. It will interact with the Core layer's services subsystem to obtain all the necessary information that it will be trying to display

3.11.1 Subsystem Operating System

Should be compatible with most operating systems

3.11.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.11.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.11.4 Subsystem Data Structures

Populating View

3.11.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

3.12 Consequence Calculator

The Consequence Calculator will take in different factors that it will then use to try and calculate different types of risks like leak rate. It will use the Core layer's Services subsystem to obtain the factor types.

3.12.1 Subsystem Operating System

Should be compatible with most operating systems

3.12.2 Subsystem Software Dependencies

Our web application will be built under C# .NET framework. For the front-end, we will use Telerik & Kendo UI, Angular Framework, and Razor Pages.

3.12.3 Subsystem Programming Languages

C#, CSS, HTML, LINQ

3.12.4 Subsystem Data Structures

The Consequence Calculator will just take in data and calculate risk but will not store any data.

3.12.5 Subsystem Data Processing

Access the Core layer to pull data from the database.

4 Core Layer Subsystems

The Core layer contains the business logic of the application, including the security, access, and authentication tasks. The Core layer uses the Model layer. It needs to know the business objects related to each service. The subsystems of this layer are Services and Authentication to interact with data.

4.1 Layer Operating System

This layer doesn't require a specific operating system

4.2 Layer Software Dependencies

The core layer depends on the Model Layer to get the information from the databse in a readable format.

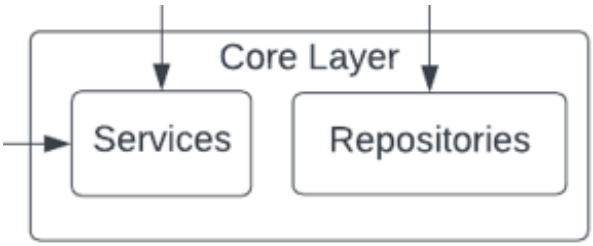

Figure 3: Core Layer Subsystem Diagram

4.3 Services

The Services are part of an implementation of the Repository pattern that basically works as a link between the .NET project and the SQL Server database.

4.3.1 Subsystem Software Dependencies

Services has a software dependency with the Business objects to be able to check what can and should be shown to the user.

4.3.2 Subsystem Programming Languages

The main language used to be able to interact with the business objects and the web applications will be C

4.3.3 Subsystem Data Structures

Must use the .net project to interact with other parts of the application so that it can display the information from the business objects.

4.3.4 Subsystem Data Processing

It will process the data obtained from the business objects through the Model Layer into a readable format for the webpage to be able to display it to the user.

4.4 Repositories

Repositories are also a part of an implementation of the Repository pattern that works as a link between the .NET project and the SQL Server database.

4.4.1 Subsystem Software Dependencies

Repositories is dependent on the .net project so that it knows how to process the data previously obtained.

4.4.2 Subsystem Programming Languages

The main language used to be able to communicate with the services subsystem and the .net project will be C

4.4.3 Subsystem Data Structures

The data structure used for repositories will be the raw data from the SQL Server.

4.4.4 Subsystem Data Processing

The data will be processed into a .json style format to be more easily read by a webpage

5 Model Layer Subsystems

The model layer, sometimes referred to as the business layer is an essential part of software architecture that divides an application's business logic from its display layer and data layer. The business layer contains the application's core functionality, such as the rules and processes that govern the behavior of the system. It encapsulates the data and provides services to the presentation layer for interacting with the data. The business layer also handles business workflows, business rules, and data validation. By separating the business logic from the presentation and data layers, the business layer promotes a more modular, scalable, and maintainable software architecture. It also makes it easier to change the user interface or data storage without affecting the core business logic of the application.

5.1 Layer Operating System

Should be compatible with most operating systems

5.2 Layer Software Dependencies

The software dependencies for the model layer subsystem include C#, SSMS(SQL Server Management Studio), and Entity Framework. C# is used to convert the tables from the database into business objects.

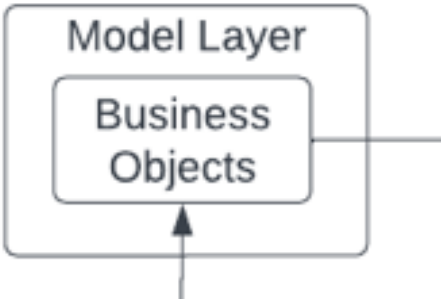

Figure 4: Model Layer Subsystem

5.3 Business Objects

The Model Layer is where the database turns into objects or a client-side representation of an object. Its main functions are connecting/accessing the database, retrieving data, and sending it back to the core layer. The subsystems of this layer are business objects. It is also being used by the Core layer.

5.3.1 Subsystem Operating System

Windows operating system is used for connecting to the database system, whereas similar results can also be obtained on other operating systems via third party platforms such as Docker.

5.3.2 Subsystem Software Dependencies

The software dependencies for the model layer subsystem include C#, SSMS(SQL Server Management Studio), and Entity Framework. C# is used to convert the tables from the database into business objects.

5.3.3 Subsystem Programming Languages

The programming language used is C#.

5.3.4 Subsystem Data Structures

A procedure knows as scaffolding mechanically converts each database table into C# class.

5.3.5 Subsystem Data Processing

The layer's subsystem is for creating business objects. Each database table will be converted into C# class that create our business objects.

6 Database Layer Subsystems

The Database layer is where all the database schema and application data are stored securely on the server side. Its functions are receiving requests from the Model layer, and sending responses including data back to it accordingly. The subsystems of this layer are Database/SQL Server, and Project Scripts and Files.

6.1 Layer Operating System

Should be compatible with most operating systems.

6.2 Layer Software Dependencies

The software dependencies for the database layer subsystem include SQL, and SSMS (SQL Server Management Studio). SSMS is compatible with Windows only. Mac and Linux users can use Azure Data Studio instead.

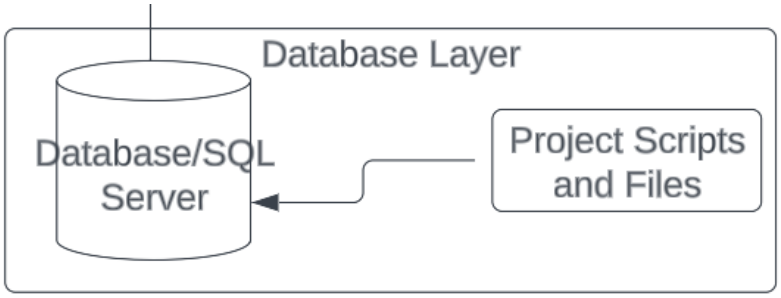

Figure 5: Example subsystem description diagram

6.3 Database

A structured repository that stores and manages a vast amount of data. It offers an organized approach to information organization and retrieval, enabling effective data storage, retrieval, modification, and analysis, supporting a range of applications.

6.3.1 Subsystem Hardware

There is no hardware specific to this subsystem because the Azure SQL Database is used for this application. The Azure SQL Database is a managed cloud database provided as part of Microsoft Azure services, therefore it is not counted as hardware.

6.3.2 Subsystem Operating System

This subsystem does not require any specific operating system.

6.3.3 Subsystem Software Dependencies

The software dependencies for this subsystem include SQL, and SSMS (SQL Server Management Studio). SSMS is compatible with Windows only. Mac and Linux users can use Azure Data Studio instead.

6.3.4 Subsystem Programming Languages

SQL

6.3.5 Subsystem Data Structures

The structured data of this subsystem is data follows a specific database schema. It is structured into tables (entities) with columns (attributes) in it.

6.3.6 Subsystem Data Processing

Every table in the database will need to be converted into C# classes and it will be done with the help of Scaffolding.

6.4 Project Scripts and Files

Project scripts and files are a collection of .sql scripts and .xlsx files used to create SQL scripts.

6.4.1 Subsystem Hardware

There is no hardware specific to this subsystem.

6.4.2 Subsystem Operating System

This subsystem does not require any specific operating system.

6.4.3 Subsystem Software Dependencies

The software dependencies for this subsystem include SQL, SSMS (SQL Server Management Studio), Visual Studio, and Entity Framework. SSMS is compatible with Windows only. Mac and Linux users can use Azure Data Studio instead.

6.4.4 Subsystem Programming Languages

SQL, C#

6.4.5 Subsystem Data Structures

A collection of .sql scripts and .xlsx files used to create SQL scripts.

6.4.6 Subsystem Data Processing

Every table in the database will need to be converted into C# classes and it will be done with the help of Scaffolding.

7 Appendix A

References