Senior Design Final Report

“Helix”

Version 1.0 – 05/13/2022

Team Members:

Dean Baquir
Norman Avery
Josh Mermelstein
Daniel Ibanez
Noe Lopez
Abdullah Alwabel
Jeffrey Lum
Jose Mierzejewski
Sameen Khan
Wilson Tobar

Faculty Advisor:

Keenan Knaur

Liaisons:

Francisco Guzman
Luis Arcantur
Julian Gutierrez

Table of Contents:

Contents

1. Introduction:

1.1. Background:

QTC is the nation's largest government-outsourced occupational health and disability examination provider. They provide medical examination and diagnostic testing services to their clients, as well as identifying and forecasting program needs. They are a part of Leidos, a $10 billion (about $31 per *person* in the US) technology solutions corporation. The organization processes over 4 million pages of medical records per day, and for efficiency, they desire an improved automated workflow solution.

1.2. Design Principles:

The web application “Helix” is used to view data files displayed by standard HTML5 compatible browsers such as Mozilla Firefox, Safari, Internet Explorer, Microsoft Edge, Opera, and Google Chrome. Helix is intended for private use by QTC and its associates. This application provides users belonging to different lines of business the ability to access specific features unique to each respective line of business. Such features include the ability to scan and view medical exam documents. All respective files will be accessible by this web application, known as “Helix”. If a user belonging to a line of business deletes a file, it will be deleted from the users’ view, however file will still exist in its original file repository.

1.3. Design Benefits:

By following the architecture provided in QTC’s coding standards, we provide QTC developers with the ability to easily expand on the application. The modular and extensible design will enable external developers to integrate their own databases, file repositories, and implement required features for their business needs. This allows for faster deployment time should QTC choose to replace their current solution, which is tight decoupled and performs much slower due to reasons explained in Section 2.1.

1.4 Achievements:

Throughout the course of the 2021-2022 academic year, our team was able to construct a generic and functional web application using Microsoft Visual Studio 2022. The web application has been redesigned for extensibility with modern back-end frameworks and leverages dynamic configurations. The user interfaces have also been redesigned with modern front-end frameworks as well as CSS styling and JavaScript logic provided by QTC, allowing the application to be simple and easy to use. Interfaces written in C# for .NET 6 have been created to implement application features associated with GUI buttons such as View, Index, Confirm, Delete, and Repair, all of which are common features of the application. The interface has been kept simple enough that it can easily be modified if needed.

2. Related Technologies:

2.1. Existing Solutions:

The current solution for QTC employs is a web application known as EFM (Exam File Manager). EFM uses classic ASP.NET inline logic to obtain medical records for each line of business. EFM is not extensible nor scalable, and lacks the ability to add, delete, or modify

features without affecting the entire application. EFM also lacks the flexibility to add new lines of business. The scalability of the existing solution is limited by its design which draws on “tight coupling” design patterns.

2.2. Reused Products:

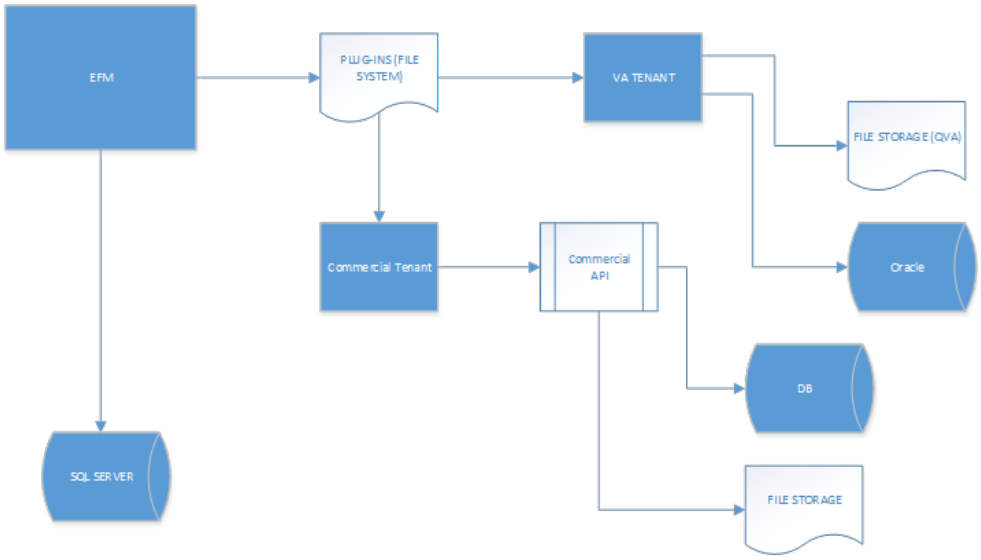
The architecture and framework used in this project was provided by QTC. The project is divided into multiple layers, each providing various sections for the Model-View-Controller (MVC) pattern. Styling for page elements as well as scripts used by these pages that handle the Document Object Model (DOM) are created to be reusable throughout the project and can easily be modified to handle future requirements. Styling components and scripting patterns were provided by QTC. Coding environments, Visual Studio 2022, SQL Server Management Studio and Oracle DB are reused to build the application.

3. System Architecture

3.1. Overview:

The framework used for this project was provided by QTC. The logic and code requirements for each line of business are intended to be developed in an isolated environment and compiled into an assembly plugin. The application is designed to load all assembly plugins at program runtime. Common feature implementations are encapsulated and injected into application objects for each line of business. These objects are known as “tenants,” and are instantiated once the assembly plugins have been scanned and loaded by the application.

Each tenant can interact with other systems that are specific to their needs, according to programmed logic compiled in the respective assembly plugin. Such systems include external file repositories and databases. If there is a need to modify or debug code for a line of business’ tenant object, the plugin can be recompiled and replaced in the by the web application without affecting other tenant plugins. The following diagram portrays how different systems interact with the web application.



3.2. Data Flow:

3.2.1 Database:

The data necessary to populate the view is stored in a database.

3.2.2 Back End:

Manipulates business logic with reference to the database, making it suitable to populate the "processed" or "manipulated" output for the views.

3.2.3 Front End:

The visual aspect of the website that interacts with the user after the data (derived from the database) is manipulated via back end.

3.3. Implementation:

The project can be summarized by two distinct characteristics being "features" that can be implemented, and a "database."

3.3.1. Features:

The web page serving medical record document files main features that can be implemented by interfaces described previously in this document:

- **Complete**: Allows the user to complete the page indices for the medical record file. Helix passes the action and file reference from the font end to the tenant object responsible for implementing this feature. An example is with the line of business known as "Veteran Affairs", or VA for short. The tenant object inserts a record into a database table that is used for queuing the process new pdf document creation based on the indices of the medical record.

- **Index**: Allows the user to enter the page indices for a specific specialty. Helix passes the input from the UI form to the tenant object responsible for implementing this feature. For VA Example, this tenant will update the MR_Index table with respective indices.

- **View**: Allows the user to view the file. Helix passes the file reference to the tenant object, which retrieves the medical records file from the file repository and returns a file stream object back to the front end.

- **Repair**: Allows the user to repair an existing medical record file. The tenant is responsible for this feature. Helix will only pass the account file reference.

- **Delete**: Allows the user to delete an existing file from future view. The tenant is responsible for this feature. Helix will only pass the account file reference to the tenant object.

3.3.2. Database:

The backend logic was designed to interface with a database running on Microsoft SQL server framework and Oracle. This database consists of several tables, containing file metadata, file locations,

document-keyword association, and document-category association. The databases schemas also contained several stored procedures. Helix will call these stored procedures to retrieve the relevant file paths.

4. Conclusions:

4.1. Results:

The web application known as “Exam File Manager” that QTC currently uses has been successfully redesigned to be extensible. With the new design, new lines of business can now be easily added to the application, respective features can be implemented, and the respective logic in its entirety can be compiled into a single assembly plugin known as a “dll”. The application will scan and load the “dlls” at runtime. This extensibility allows for new features to be implemented by line of business without affecting other lines of business. The Search, View, Index, Confirm, and Delete features have all been successfully implemented for the Medical Records page respective to the “Veteran’s Affairs / VA” line of business.

Some features for the VA line of business are pending implementation, such as the functionality for the other UI tabs meant for different record types, such as Signed Reports and Signed 201 for example. In-line logic previously used for retrieving data has been successfully replaced and moved to SQL stored procedures.

4.2. Future:

The application was made with the goal of increasing productivity for QTC. The generic implementation allows different lines of business to be able to update their respective application logic to meet their unique needs. This allows for decoupled, dynamic, and extensible development. Future improvements that could be beneficial, but were not considered during the project’s development cycle include the following:

- Real-time queues. Currently, the system retrieves data from a database and displays it as a static page. If too many users log onto the system at the same time, the network could become congested with requests and responses from the database. This can be resolved by implementing SignalR which is compatible with ASP.NET.
- User roles need to be properly implemented into both the application, and the SQL database for the application to work with all pages.
- Creating general purpose components for use in the Network and Database subsystems would improve reusable code and cut down a lengthy development process. By creating more store procedures and triggers, developers can spend less time on running efficient payloads, regarding both requests and responses with payloads.
- Expanding to a fully generic queue for distinct types of workflows.

5. References:

- Visual Studio Documentation: https://docs.microsoft.com/en-us/visualstudio/windows/?view=vs-2022&preserve-view=true
- SQL Server Management Studio Documentation: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15