System C Support App – Project Outline

Introduction:

This project was completed as part of a 12-week summer internship from June – September 2023. This project aimed to build a web application to assist the SystemC customer support team.

System C provides health information technology systems and services to the NHS and social care. One of their services stores and manages patient and baby data for maternity and neonatal care. This data is stored in the form of entity records that have their own GUID (globally unique identifier) as their key identifier. When the hospital-based clinical users of this service encounter problems, they can contact System C, and the support team can help to diagnose the issue. This is where the GUIDs are used to look up the relevant patient care record data, without sharing patient information over the phone. The previous method for the support team to look up any data/information was not user-friendly and often required a skilled engineer to find the root problem. This web app aims to create a more functional and user-friendly tool to help the support team fix problems quickly.

Disclaimer:

All the data that is shown in the figures below is simulated and anonymised due to data protection requirements.

Technology Stack:

Blazor was used as the framework for this project due to its ability to create dynamic web applications using only C# and HTML. Another benefit of Blazor is being able to design generic components which can be used throughout the project, which means it is very quick to implement new functionality. The front end was styled using Tailwind CSS, which is a utility-first CSS framework.

Figure 1 - Technology Stack

Git, as part of the Azure Dev Ops platform, was used to manage the version control of the project, with work being done in branches of the main company repository. The project was deployed using Microsoft Azure.

Select Entity Page:

When the user navigates to the application, they must input a GUID. Once entered, a check is completed to determine two facts:

- Is the input an actual GUID in the correct format?
- Does the GUID have a corresponding entity in the database?

If the GUID passes both checks, then the user is brought to the summary page (Figure 2).

Summary Page:

The navigation bar on the left now displays the different functionality that comes with the chosen entity (more detail is provided on this in the 'Next Steps' section below). The following pages are templates where the data that is shown is generated using the provided GUID. The GUID is passed to a Web Service which is connected to the company database, which returns a JSON string. This string is used to populate the web page with information. This summary page (Figure 2) shows some basic information about the entity, and any relevant graphs and figures.

The benefit of using Blazor components is introduced here. In the figure below, the coloured 'blocks' which in this project are referred to as 'pageblocks' have been designed to be as generic as possible. Their colour, headings, content, height, width, scroll options, etc. are all properties of a C# class. This means the returning JSON string can provide all the needed information to generate any combination of these 'pageblocks'. Therefore, new pages can be created quickly with simple tweaks to the JSON, with no new code being written.

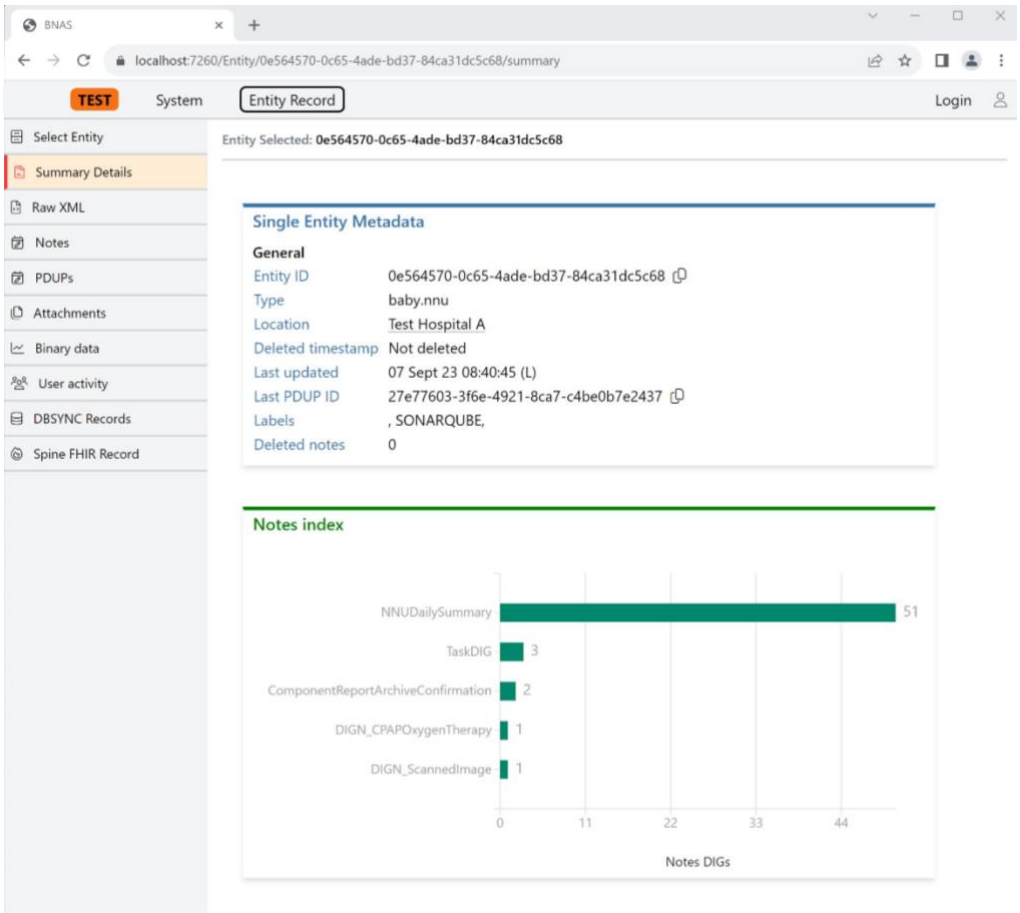

Figure 2 - Summary Page

Raw XML Page:

The data for each Entity Record, which represents a single patient care record, is stored and transferred as XML files. It is therefore useful to view the raw XML file that is related to each entity to find any errors in the actual data. This page (Figure 3) uses the Blazor Monaco NuGet package, which renders functional code editors in a range of languages. The raw XML from the database is passed into the editor, and automatically, the syntax is highlighted, and the code is indented and collapsible.

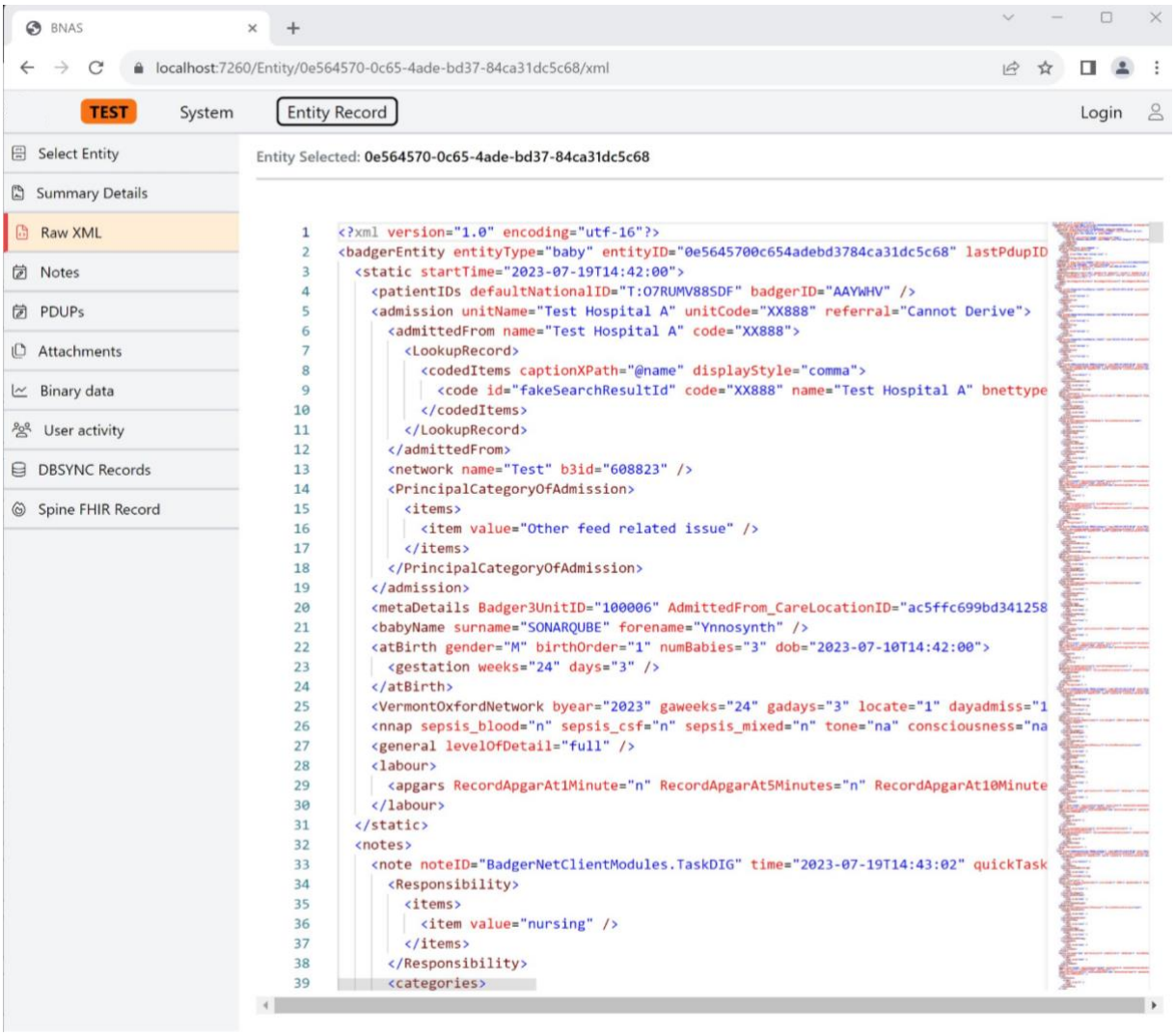

Figure 3 - Raw XML Page

PDUP Overview Page:

When changes are made in the database to any patient (entity), this is referred to as a PDUP (Patient Data Update Packet). It is important that a record is kept of all these updates. This page (Figure 4) shows a summary of all the PDUPs related to the selected entity. It shows again how the generic ‘pageblocks’ are used in this project.

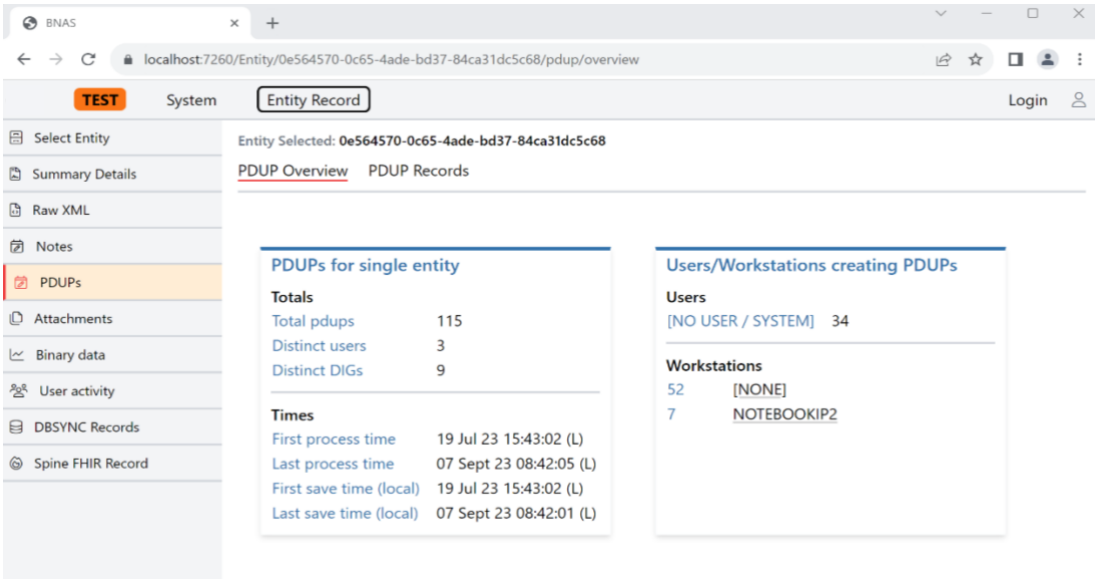

Figure 4 - PDUPs Overview Page

PDUP Records Page:

This page (Figure 5) shows a list of all the PDUPs for the chosen entity. The list on the left is another example of using Blazor components, which means it can be repeated throughout the project by just changing its input. When one of the PDUPs is clicked by the user, it updates the summary details on the right. This shows all the relevant information of this PDUP in the form of a 'pageblock'. The 'View Raw PDUP XML' button creates a pop-up module with a Blazor Monaco code editor with the raw XML of the selected PDUP. The filter button was not completed by the end of the internship but was the next feature to be completed.

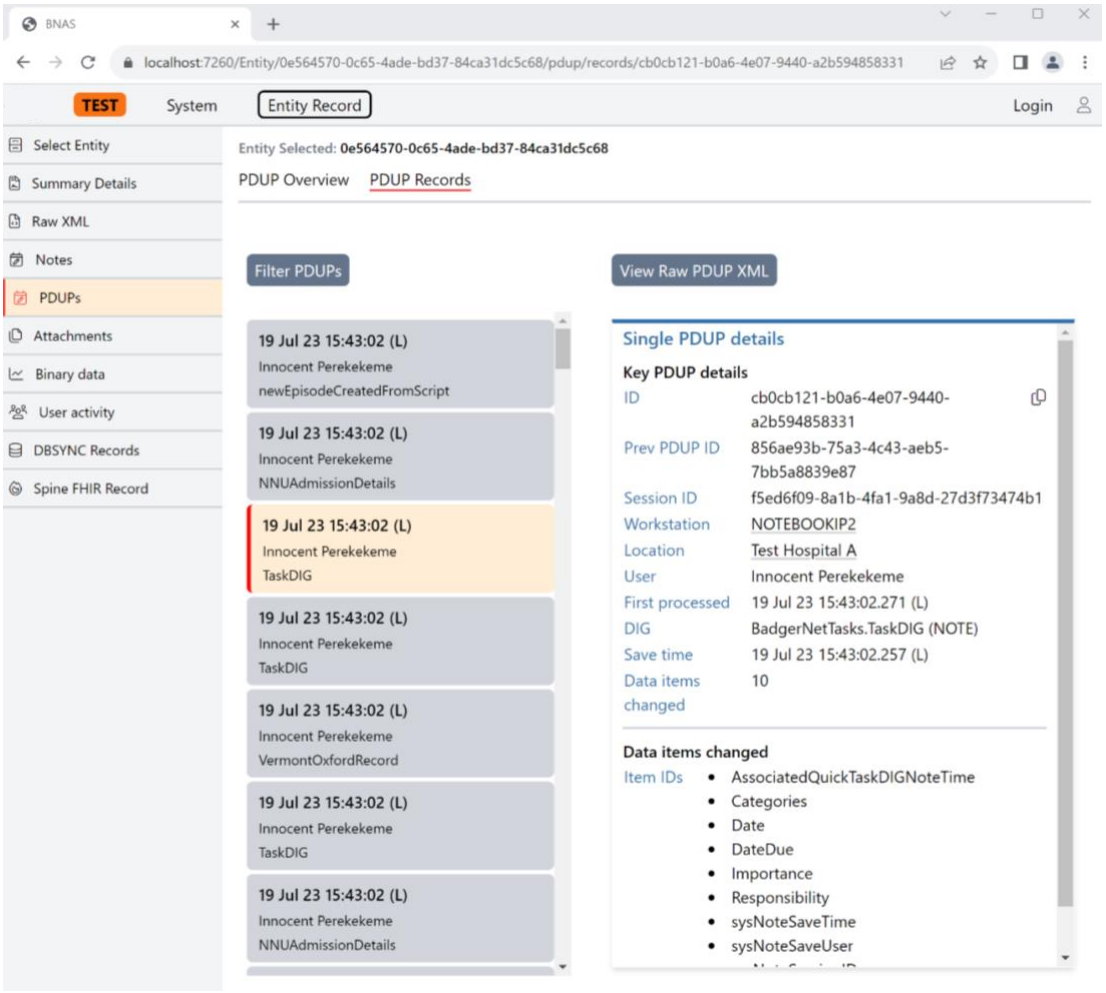

Figure 5 - PDUP Records Page

Attachment Page:

The production Care Record system solution also provides the ability to store attachments related to patient care record entities. These are usually in the form of pdf/png/jpg files. This page (Figure 6) shows a list of all the attachments related to the selected entity (this is the same list component as the PDUP page). When an item is clicked on by the user, the 'pageblock' on the right updates with a summary of the attachment. The 'View Attachment' button displays a pop-up module with a html iframe, which shows the attachment (Figure 7).

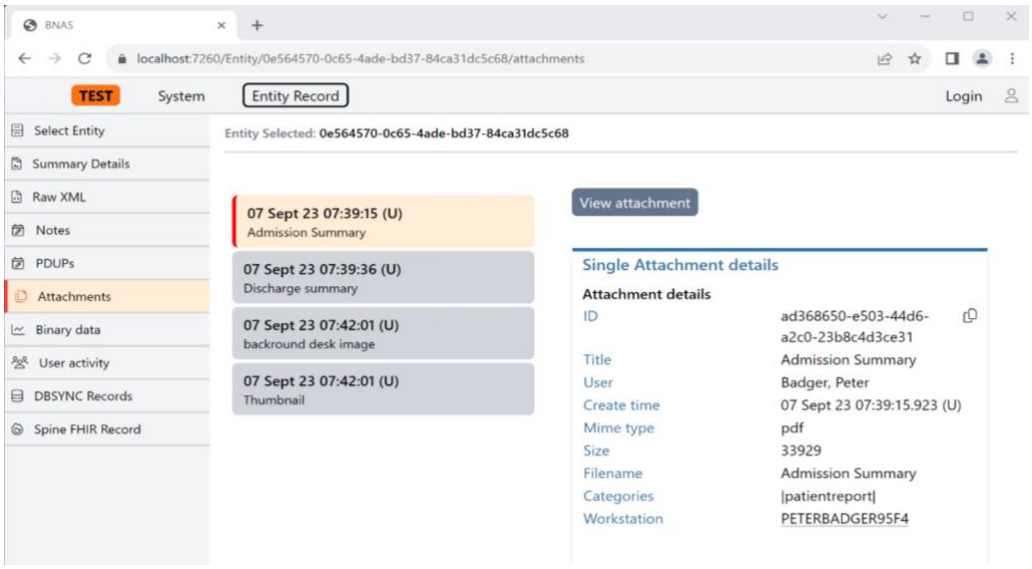

Figure 6 - Attachment Records Page

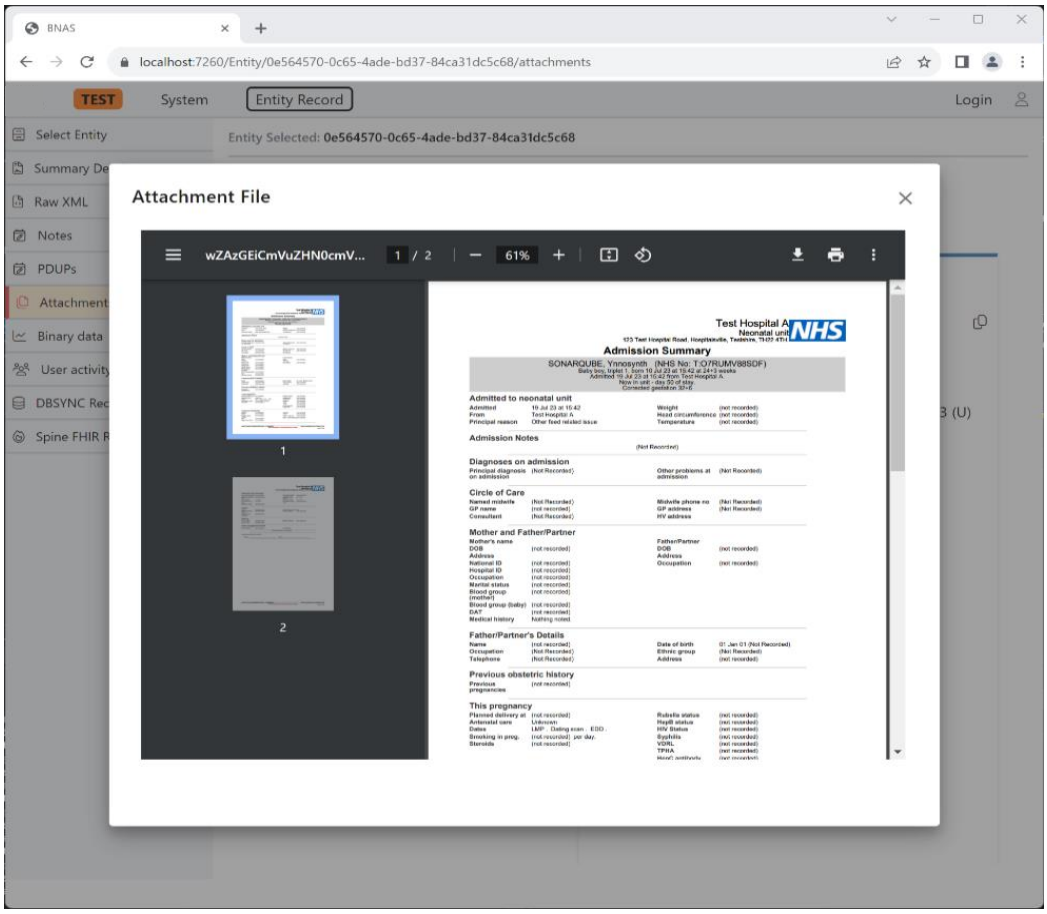

Figure 7 - Attachment Pop-up Module

Next Steps:

This project was not fully feature-complete by the end of the summer internship, but some of next functions to be added are:

- A filter button for the lists of PDUP results, since there could be hundreds for each entity, being able to filter these is crucial.

- Audit trails of actions by each user.

- Include user authentication using Azure authentication. This would give the option for different functionality access for different roles within the company.