The software is a classic ASP.Net Core MVC application, but can be somewhat bewildering to the uninitiated, so this walk through is designed to give a brief overview of the way the application runs.

The application makes heavy use of the jQuery **Datatables** plugin for rendering the table grids in the application, so there is a lot of client side java script to enable this, so sometimes application can feel more like a Web API application than traditional MVC app.

**Application Startup**

When the application is started on the server,a **Program.cs** class is instantiated and the program entry is the **Main()**

Method. Together with the startup class **Startup.cs** a webhostapplication is configured and built with configurations set in these classes, and the application starts listening for http packets.

During this build process a middleware pipeline is configured, services are registered with the dependency inversion container, and the kestrel HTTP server is started.

**User Path through Software:**

The entrance to the program is **home/Index** this is the default path

Using MVC conventions means that action methods are called on Controllers:

Controller=HomeController Action Method =Index returns View **Views/Home/Index.cshtml** torender the Index razor view.

This is the usual convention for stating an MVC app.

The user must be authenticated before given access to the database, so the HomeController class is decorated with the [Authorize] attribute, so if user is not authenticated by the Identity Middleware system they will be passed to back to **Account/Login** page

Controller=AccountController Action=Login View View=**/Views/Account/Login.cshtml**

When a user is authenticated by Username and Password then opens view :

**Views/Home/Index.cshtml**

The starting view is created with a series of partial views

This Index view itself does not do very muchit runs a javascript **Patients.BindPatientsModals** scriptand then renders:

**Views/Patients/Index.cshtml .**

All the views are rendered with a layout view **/Views/Shared/\_Layout.cshtml**

This **\_Layout is** fundamentalto the application and sets the template for the database layout for the whole application including headers, footers, Navigation etc and defines how the application will appear to the user.

The view **Views/Patients/Index.cshtml** is renderedwithin the layout at the directive:

**@RenderBody ()**,

The **a Views/Patients/Index.cshtml** isitself is composed of anumber of partial views**,** thekey one being**: Views/Patients/\_Datatable.cshtml**

Thisis where the html table called **patients\_datatable** is defined**,** which has an html id of **patients\_datatable** and this the one operated on by the client side **DataTables** plugin and the javascript defined in **wwwroot/js/Patients.js** which renders produce the main patients grid Patients page within the layout.

**/Views/Shared/\_Layout.cshtml** has a number of other important placeholders as well as the **@RenderBody** directive, it defines a placeholder for the bootstrap modal forms popup forms which are used for data entry and editing

**<div id=”modal-container”>**

Also this \_Layout another partial form is rendered:

**@{await Html.RenderPartialAsync(“\_Scripts”),}**

This partialview **Views/Shared/\_Scripts** is just a long list of javascript files which are run to initialize the client side javascripts needed by the app.

These scripts are found under the project **wwwroot/js**  folder.

Most the scripts are written using the javascript **module pattern** (self invoking closures).

The main one for setting up the Patient Tables is **wwwroot/js/Patients.js**

At the end of the **\_Layout view** is a **RenderSection** with another jquery script:

<script>

$(document).ready(function () {

pageSetUp();

drawBreadCrumb();

App.init();

});

</script>

The **App.Init** method code is:

init: function () {

onLogoutLinkClick();

Patients.init();

Settings.init()

Drugs.init();

Users.init();

Imports.init();

DiagnosisTypes.init();

SideEffects.init();

Charts.init();

Search.init();

Radiology.init();

PatientVisits.init();

CaseReportForms.init();

AnonPatients.init();

MedicalTrials.init();

AllergyIntolerance.init();

UI.init();

Reports.init();

}

**App.Init** is the method in **App.js** API that first Initializes some of the basic infrastructure

The **Views/Patients/Index.cshtml** has a number of nested partial views to render

the

**@Html.Partial("~/Views/Patients/\_Buttons.cshtml") @Html.Partial("/Views/Patients/\_DataTable.cshtml")** is where the patient grid **Patients\_Datatables** is

Defined

The three buttons inside the grid are defined by the **initPatientsDataTable** in the **Patients.js script**

The 3 rendered buttons are identified by classes:

**patient-details** event handler set by **bindPatientDetailsShow** function in **Patients.js**

**patient-edit** event handler set by **bindPatientEdit** function in **Patients.js**

**patient-delete** event handler set by **bindOnDeletePatientClick** in **Patient.js**

The event handlers for these buttons are set up by**bindPatientsModals** function, part of API of **Patients** module (**Patients.js**)

This function is called in a number of places including the **initComplete** property of **initPatientsDataTable** function

The **patient-details** and **patient-edit** both open Bootstrap Modal Windows using code like:

var bindPatientEdit = function () {

$(document).off("click.patient-edit").on("click.patient-edit", "a.patient-edit", function (e) {

LoadingIndicator.show();

e.preventDefault();

$.get($(this).attr("href"), function (responseHtml) {

LoadingIndicator.hide();

$("div#modal-container").html(responseHtml);

$("div#edit-modal").modal("show");

The href attribute is read, in the case of **patient-edit** is **:**

href="/Patients/Edit/' + patient.id '"

Where Patient.id is the id of patient being edited.

These buttons all use jQuery Ajax calls to get data back asynchronously to populate the modal popup forms to add or edit patient data.

So $.get($(this).attr("href"), function (responseHtml) is an Ajax call to Controller **PatientController** Action Method **Edit which** returns the html for the modal window.