

BRIT Systems

**WCTC BIT- HIT**

**Spring 2017 | Phase 1**

**Developer Documentation**

**Created On: 5/15/2017**

**TABLE OF CONTENTS**

**Table of Contents 2**

**Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_3**

**Key Purpose(s) of System \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_4**

**Phase One Development Scope \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_5**

**Key System Information \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_\_\_6**

**Security Test \_\_\_\_\_ \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_6**

**Data Model \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_7**

**The Future \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_8**

**BIT Team Member and Faculty Information\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ 9**

**Software Architecture \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ \_\_\_\_10-end**

**Introduction**

BRIT Systems is a web application built as a result of the collaboration between the BIT (Business Information Technology) Web and Software Developer and HIT (Health Information Technology) programs at Waukesha County Technical College in the spring of 2017 (January-May) .

This document has been created to document the key facts about the development of the application. All information within this document is accurate to the best of the documenter’s ability on the date of submission, 5/18/2017. Given the nature of the application, it is very likely that changes will be made to the system after the time of submission of this document and any information found within this document should be verified as being up-to-date and accurate prior to taking action or making any decisions regarding the development and maintenance of the application. The documenter nor the BIT students named later in this document claim any responsibility nor obligation for the application.

Identifying a need for HIT students to have a software application which allowed them to practice entering mock Birth Records into a hospital database, the HIT faculty approached the BIT facility to request the creation of such an application

The project served as the final capstone project for the participating BIT students, of which there were four, as well as the two participating HIT students. While the project was introduced, monitored, and verified by faculty from both programs, the project was generally lead and executed by the students from the programs.

This document is intended to be a high-level overview of the system development during phase 1. Any questions regarding the development of the application should be directed to BIT faculty.

It is important to note that BRIT Systems is meant to house “mock” data and not real patient health records. It is not HIPPA Compliant.

Going forward, the application will be maintained and further developed by future BIT students and faculty.

.**Key Purposes of System**

The key purpose of BRIT Systems is to provide a tool which may be used in conjunction with HIT coursework to allow students to practice entering electronic birth records into a database system in a manner which is similar to what is currently found in the healthcare industry.

The birth record information required to be entered in the system comes from the legal government form which health care providers are required to complete for births.

Other key requirements included the ability to print a “Souvenir Birth Certificate” and an access system which provides role- based log-ins. HIT instructors need visibility to what students users are doing in the system, and general system logging were required by the instructors.

While there were many other features which were identified as “nice-to-haves,” the requirements outlined here comprise the key purpose of the system in Phase one.

**Phase One Development Scope**

Much time and effort by both the HIT and BIT students was spent in determining what the scope of this phase of development should be. Setting an appropriate scope proved to be the key challenge in the early development phases of the application. Both parties see many potential additional uses for the application beyond what was delivered in phase 1, and it is the hope of both parties that the application will gain additional functionality.

Currently, BRIT Systems has delivered the key system purposes as defined in the “Key Purposes of System” section of this document.

Users are required to login to the application with usernames and passwords. Users may be assigned various roles, including disabled, data entry, Instructor, and database administrator.

Users may create patients, and add complete birth records to female patients. Users may search for patients by MRN (Medical Record Number), last name, or last name and date of birth. Users may print souvenir birth forms for child patients.

A secure log in system has been created which allows anyone to create a “disabled” account, which only allows users read access on patient records. Instructors, who have full CRUD and system access must set user roles to “data entry” if a user should be allowed the additional CRUD access of create and edit.

Additionally, basic system logging has been provided and includes basic details about actions users take in the application, such as account creation, log –ins, and execution of basic CRUD methods.

**Key System Information**

The application was developed as a Visual Studio ASP .NET MVC 4.5 application. The framework was chosen due to the fact that both students and instructors were familiar and comfortable with both the framework and the framework’s ability to meet the requirements in the time allowed for development. The application utilizes ADO.NET Entity Framework to communicate between the program and the SQL database.

The codebase is stored in a private instructor-provided GIT repository. Requests for access to the repository should be directed to BIT instructors.

Due to a delay in obtaining access to campus servers to host the application, the application is currently hosted on an instructor-provided Azure account. As such, the application is publically accessible, and may be found at:

<http://britsystems.azurewebsites.net/>

It is possible that the hosted location of the application will change after the submission of this document, in which case the relevant information here should be updated.

**Security Test**

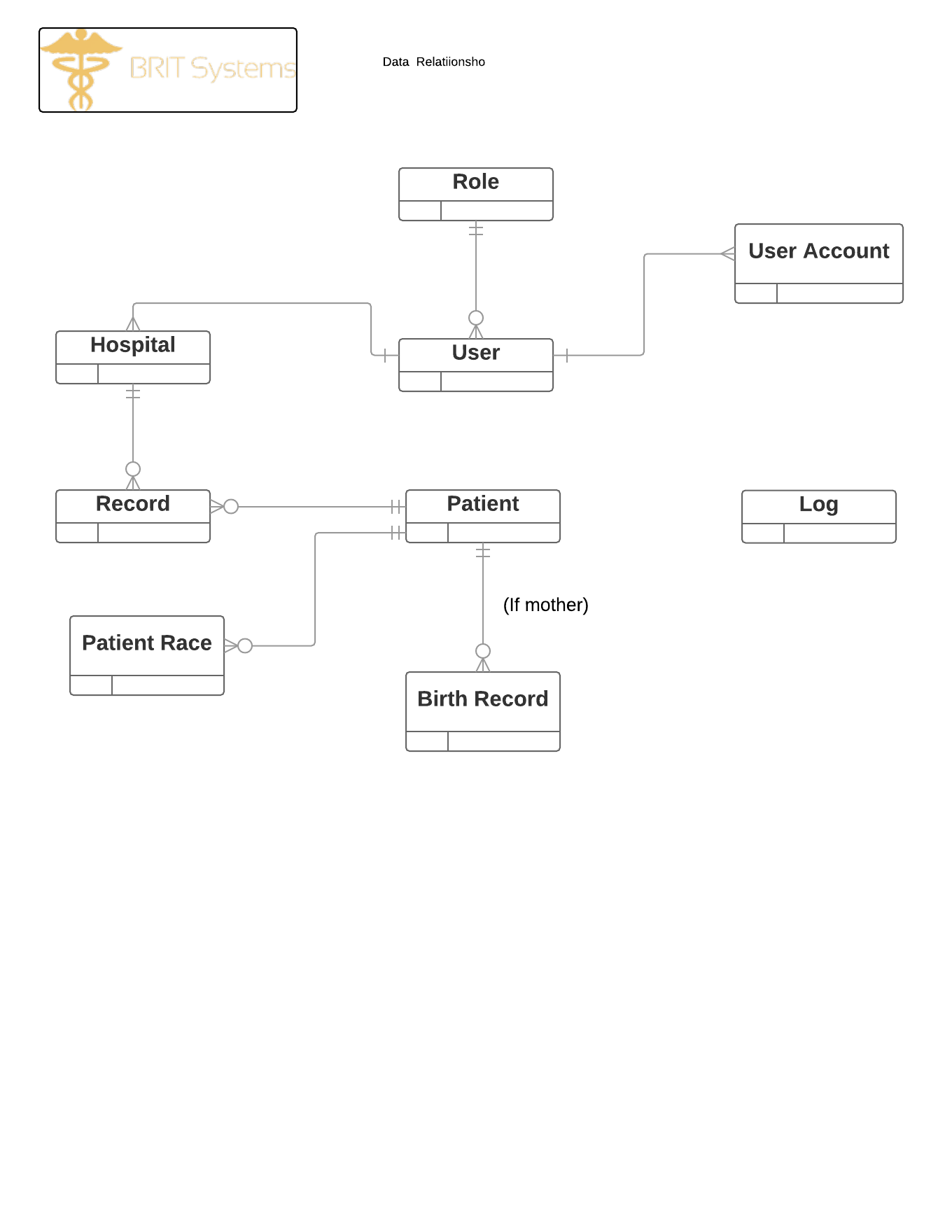
In the final week of development, a security analysis was completed by WCTC Cyber Security students at the request of the application development team. The results were generally favorable as to the security of the site, with a few identified concerns.

One important vulnerability was identified and resolved. Due to weak password requirements, security testers were able to guess a user’s login credentials and gain illegal access to the system in that manner. After this discovery, password requirements were updated to require that passwords be at least 10 characters in length and include uppercase, lowercase, and number characters. HIT students assume no responsibility for the current or future security of the application.

Anyone completing future development on the application should request the full security report provided by WCTC Cyber Security students from BIT faculty instructors.

**Data Model**

Particular attention was paid to developing a data model which not only meet the need of phase one, but one which also could support the creation of new and more complex table relationships into the future.

****

**BIT Team Member and Faculty Information**

**Faculty Instructors**

Andrew Kjell

Matthew Green

**Student Developers**

Michael Haydon

William Boyer

Anthony Mellem

Jennifer Scheidegger

**The Future**

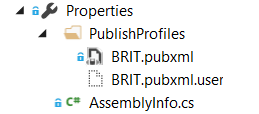
The BRIT Systems application was designed and developed with the hope and expectation that future BIT students would continue to develop the application and add new features and functionally as identified by facility and end user input. The developers believe they have created a data model which will be an extendable foundation off of which to base future growth.

Currently the GIT repository contains a backlog of Issues. These issues were identified and added by the BIT team members who worked on the application in phase 1. They were deemed out of scope for the time available in phase 1, and added with the expectation that the next team would address these.

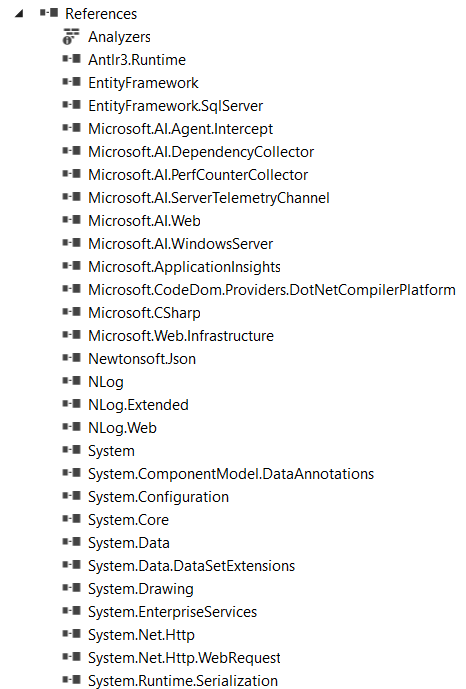
**Software Architecture- Class/Folder Structure**

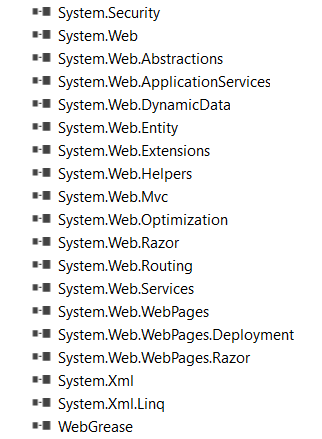
The project solution is organized into a class and file structure that is standard for a ASP .NET 4.5.2 application developed in Visual Studio. Much of the solution has been configured with the defaults provided by Visual Studio/ ASP .NET convention. Following is an outline of the current file structure of the solution, which highlights customizations.

**Properties**

****

**References**

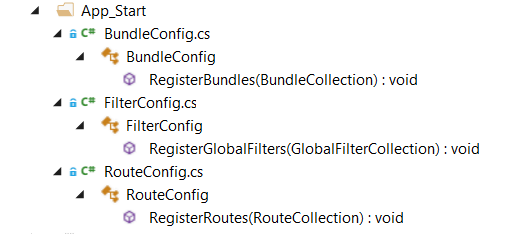
****



**App\_Data**

No files within this folder, but is required by the solution.

**App\_Start**

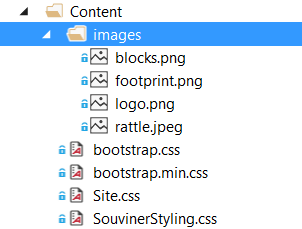
****

**bin**

The bin, or binary folder, contains files, please see current solution for file list. This is where Visual Studio stores all the complied assemblies (.dll files) for custom ASP.NET controls, components, and other code referenced in the ASP.NET web application. These assemblies are the actual executable code,

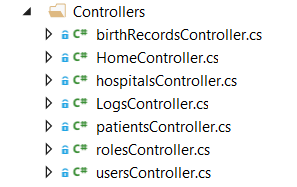
**Content**

The Content Folder contains custom images used for site design and the souvenir birth certificate form. Additionally, all CSS files for the application may be found here. Site.css has been highly customized and it used to provide the main styling within the document. SouvinerStyling.css is the stylesheet used to support the styling of the Souvenir printable birth certificate form.



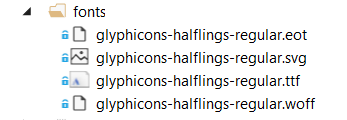
**Controllers**

The Controllers folder contains all the controllers currently available in the application. These are based off of the model classes/ data model. More information about the configuration of each controller is provided in the “Software Architecture- Class Configuration” section of this document.



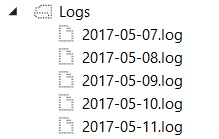
**Fonts**

The fonts folder, to the authors’ knowledge, has not been customized in any way.

****

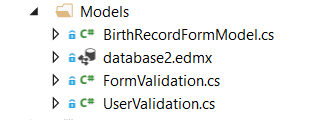
**Logs**

The Logs folder is where the debugging logs for the application are found. Logs are created on a daily basis.

****

**Models**

The Models folder contains all the data model classes currently needed to support the application. More information about the configuration of the model classes is provided in the “Software Architecture- Class Configuration” section of this document.

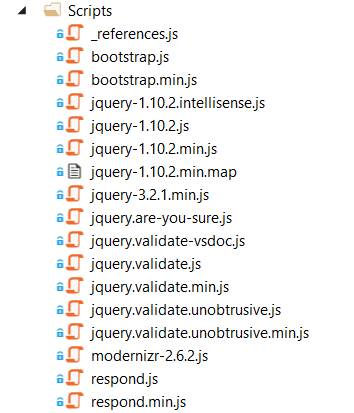
****

**Obj**

The obj folder is a default necessary folder provided by the Visual Studio tool. This folders holds **object, or intermediate, files**, which are compiled binary files that haven't been linked yet. They are fragments that are combined to create a final executable. The compiler generates one object file for each source file.

**Scripts**

Scripts is where the bootstrap, jQuery, modernizr and respond JavaScript libraries may be found. These were all included by the Visual Studio Wizard when the app was created. One custom file is found here, jquery-are-you-sure.js, which is used on the birth record forms to alert users of unsaved form changes when navigating away from a form tab. It should be noted that there is other jQuery and JavaScript code found throughout the application which may be found on the appropriate view cshtml file.



**Security**

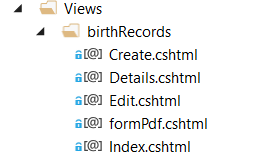
The security folder contains a custom class used for security for the user login system.

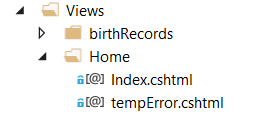
More information about the configuration of the security class is provided in the “Software Architecture- Class Configuration” section of this document.

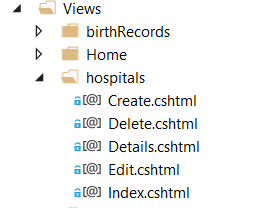


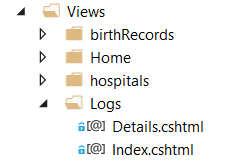
**Views**

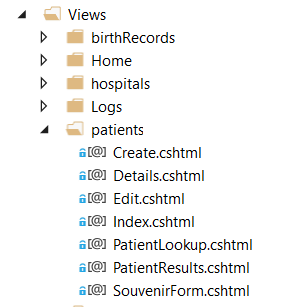
The Views folder contains all of the views (cshtml pages) currently required by the application. These views have been highly customized to meet the requirements of the project. The views are divided into folders related to the organization of the data model of the application. Further information about these views should be obtained by reading the files themselves.

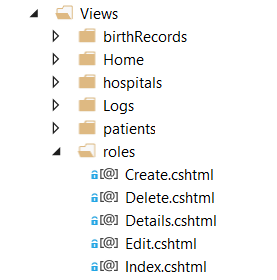


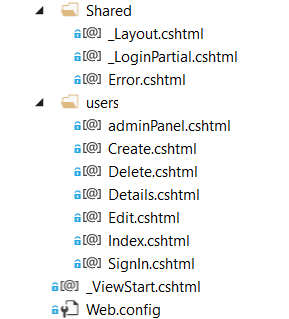






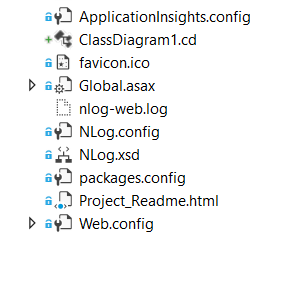






**Misc Files**

Below is pictured the reaming files found within the application, all of which have been created by the default project/solution.

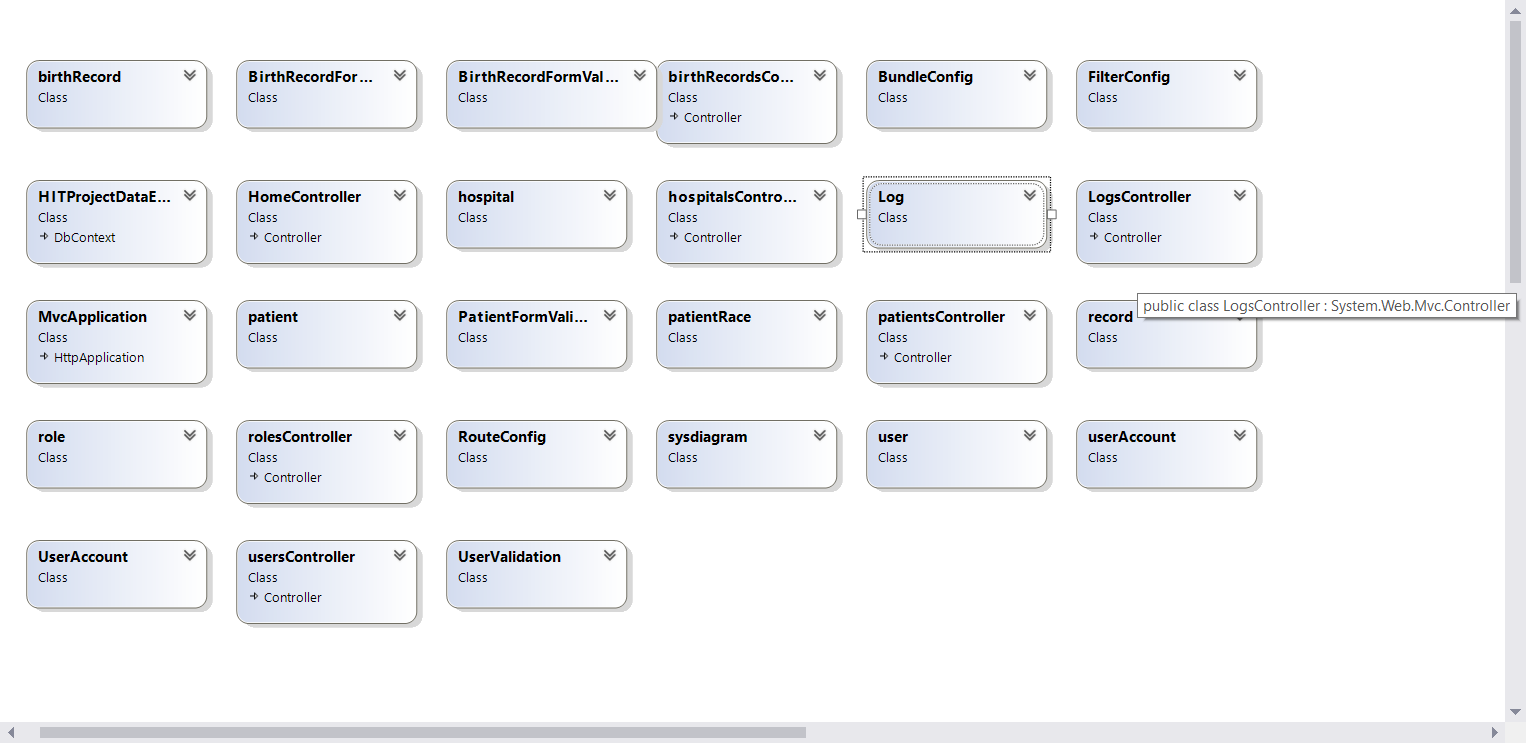


**Software Architecture- Class Configuration**

This section documents the details of the classes of the application. This includes the Controller, Model, Security, and Log classes.

**Current Class Diagram**

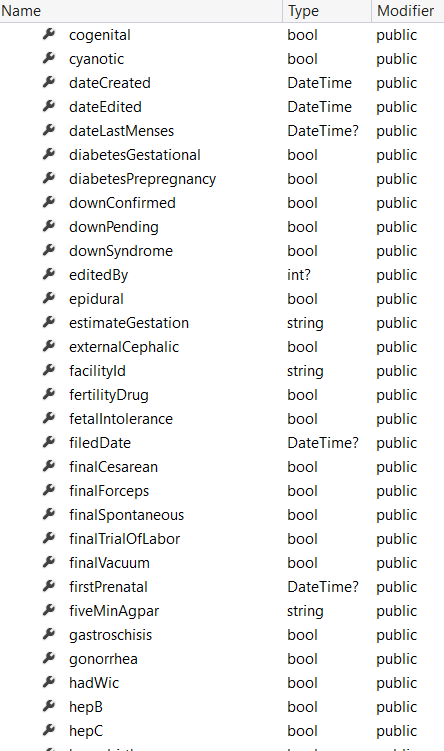
Below is a snapshot of the current .cd document which Visual Studio provides.

****

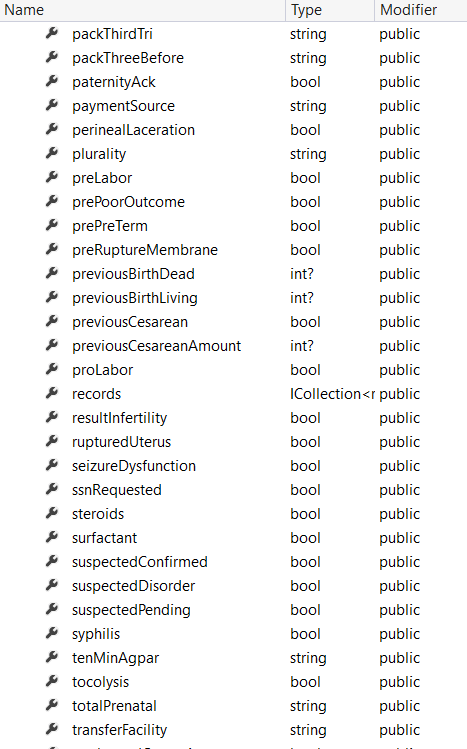
**Birth Record Class**

The Birth Record Class comprised the main part of the application and contains well over 100 fields/properties as related to the data requirements of this application.





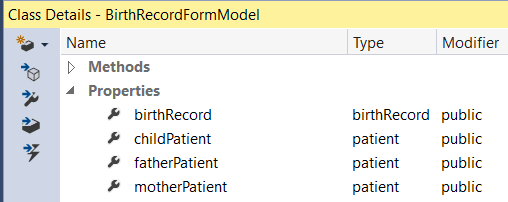






**Birth Record Form Model**

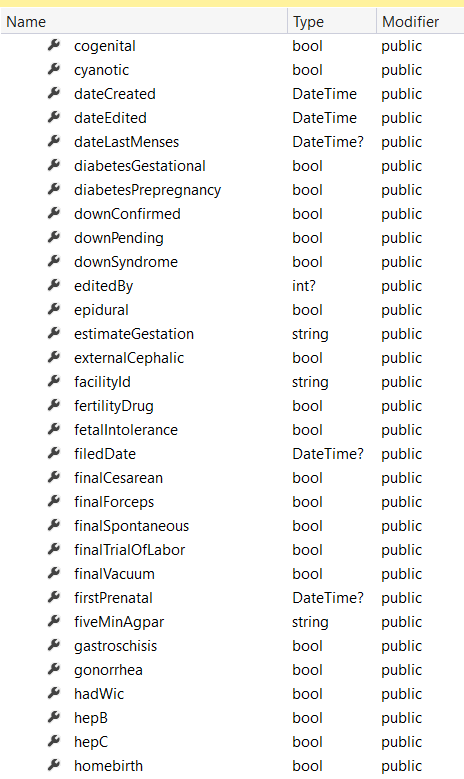
The Birth Record Form represents the birth record form and currently has 4 properties, the birth record itself, and the child, mother, and father patients.



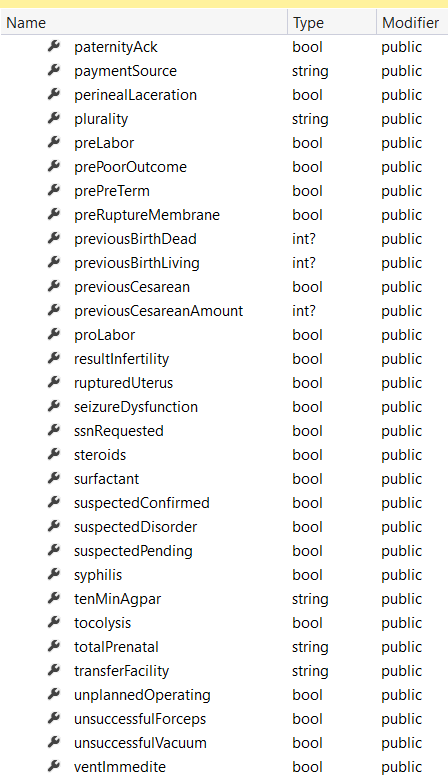
**Birth Record Form Validation**

The birth record form validation class provides custom validation on the birth record form. Per the requirements set forth by the end users, many, but not all, fields on the birth record require specific validation rules. As such, this class contains many properties.





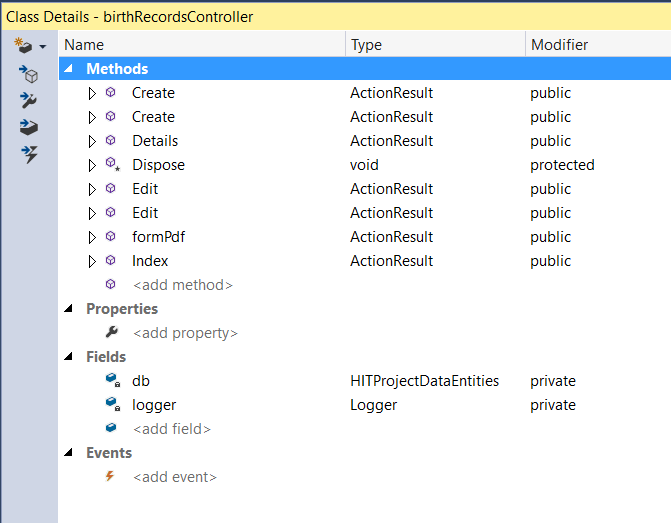






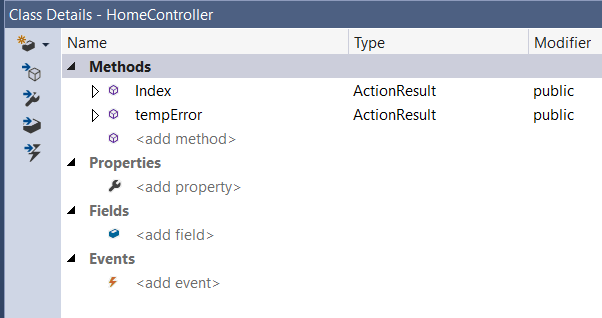
**Birth Record Controller**

The birth record controller class contains the expected controller methods.

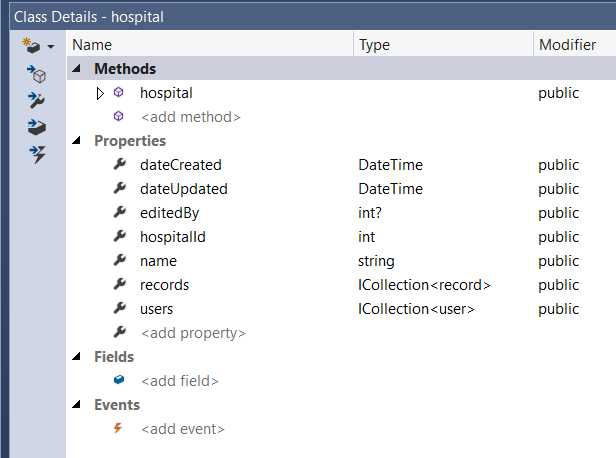


**Home Controller**

The home controller class contains the expected controller methods.

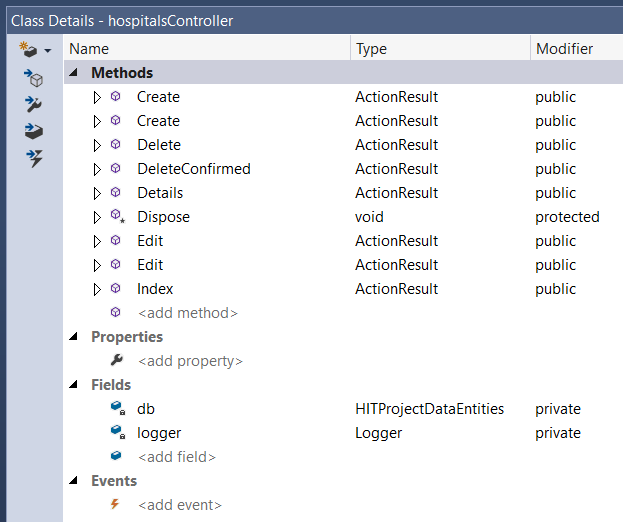


**Hospital**

****

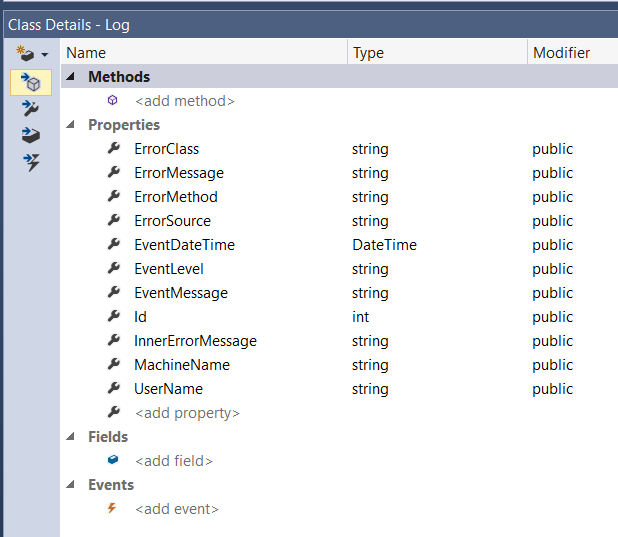
**Hospital Controller**

The hospital controller contains the expected methods. A hospital may represent a school or a class.

****

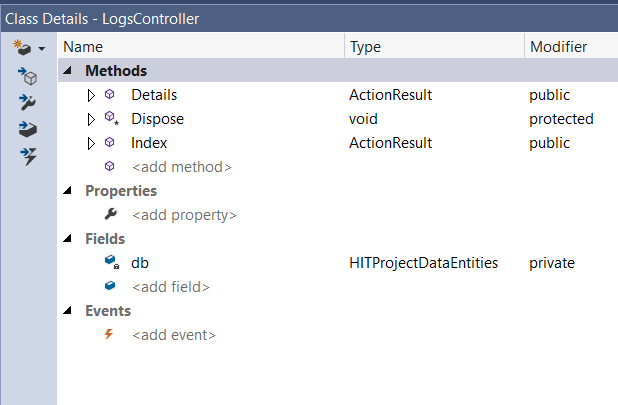
**Log**

The Log class contains the properties needed to perform system logging. Currently simple events such as user login, creation, and record CRUD events are logged.



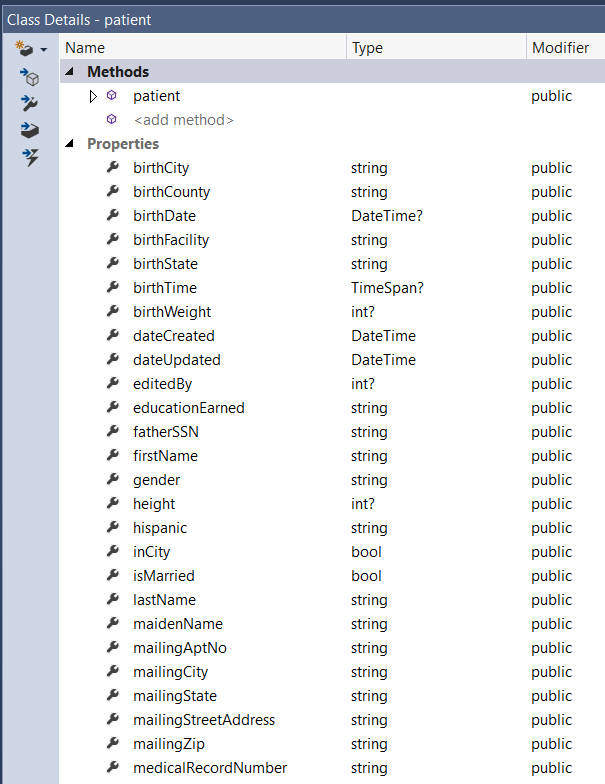
**Log Controller**

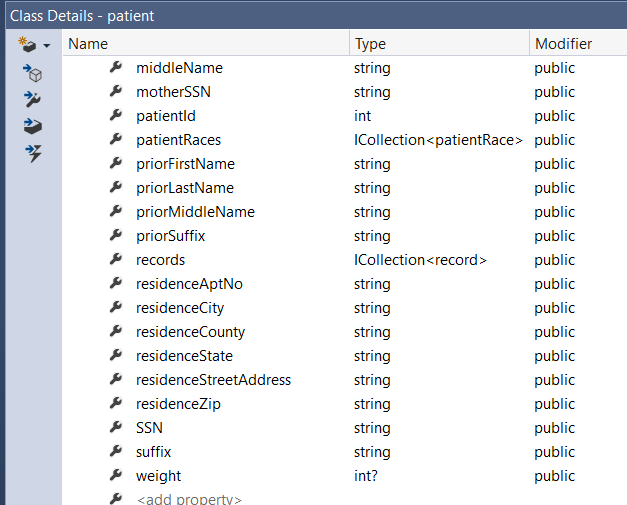
The log controller contains the expected methods for a controller related to logging.



**Patient Class**

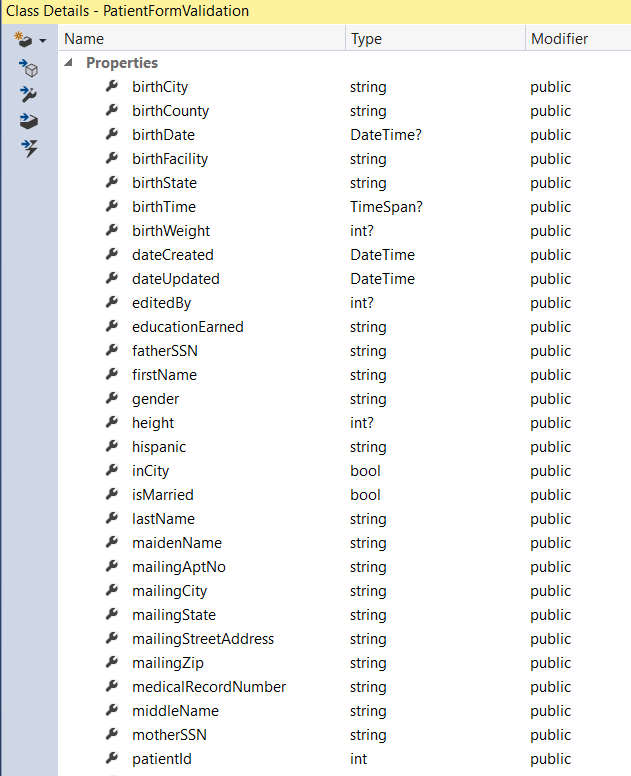
The patient class contains the properties as indicated by the data model for a patient record. All individuals, whether a child, mother, or father, are created as patient records.

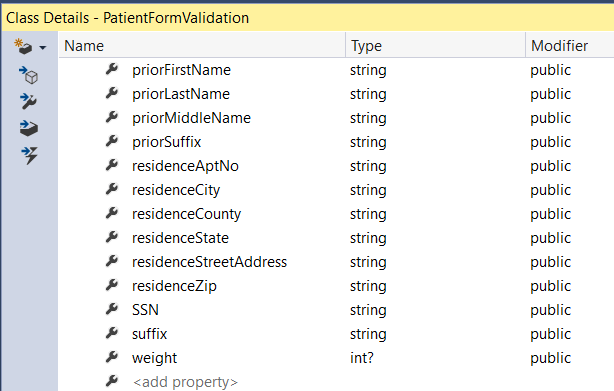
****

****

**Patient Form Validation**

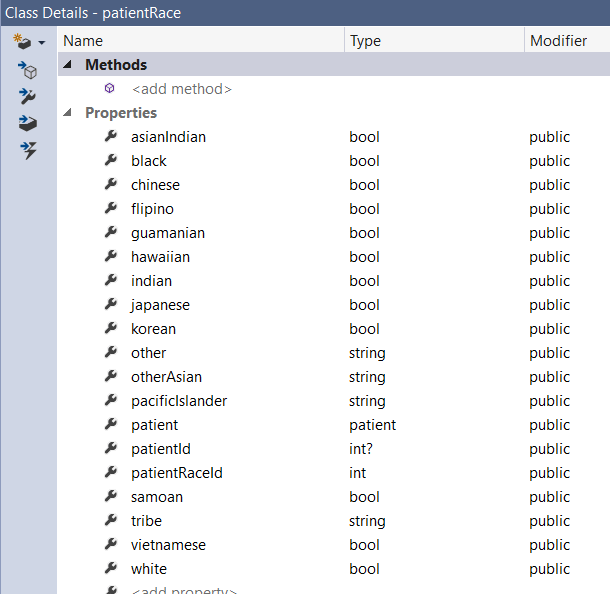
The patient form validation class contains the properties required for validation requirements on the patient form.

****

****

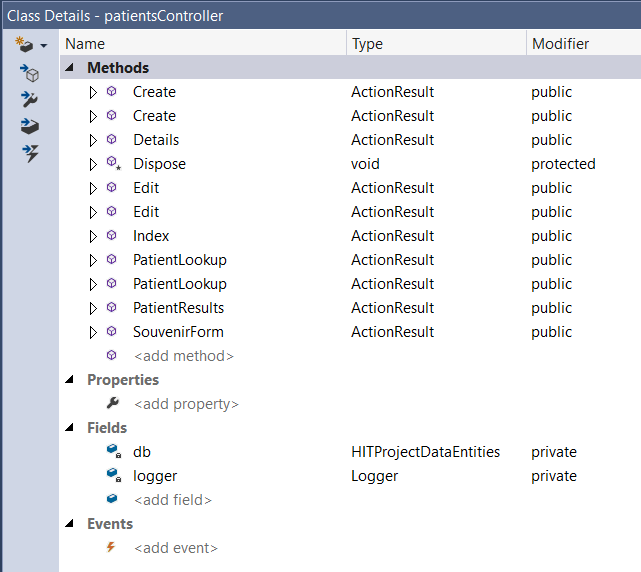
**Patient Race**

The patient race class contains the properties as indicated by the patient race table in the database.

****

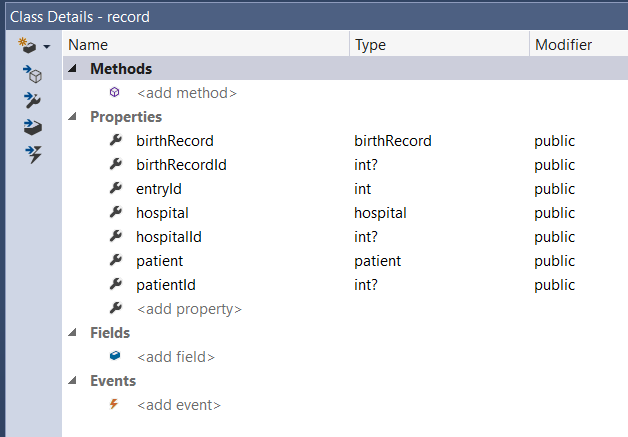
**Patients Controller**

The patients controller contains the expected controller methods and additionally contains key application specific methods related to looking up patients and retrieving data, including data to create a souvenir birth record.

****

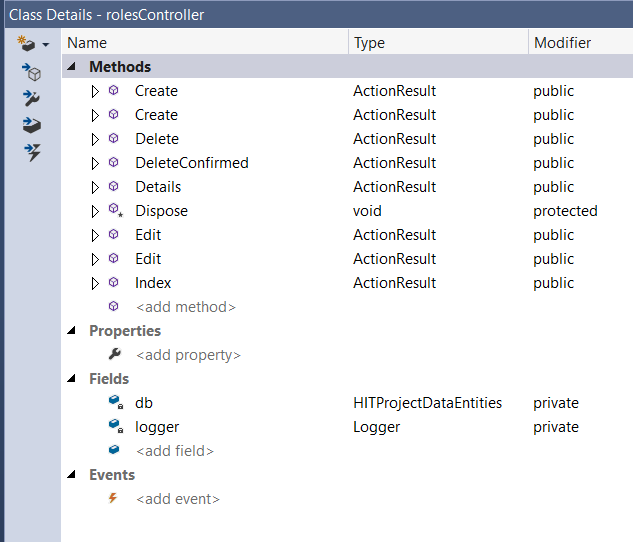
**Record**

A record represents a patient encounter with a medical provider. This class contains record specific properties.

****

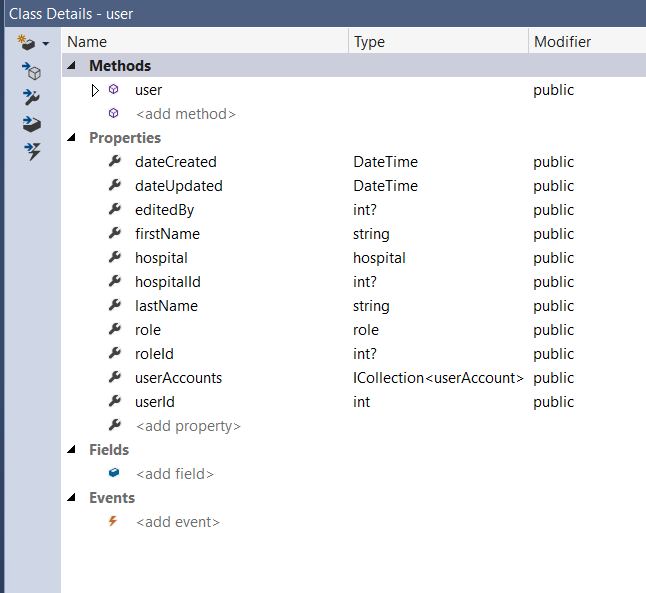
**Roles Controller**

The roles controller contains the expected controller methods related to working with user roles.

****

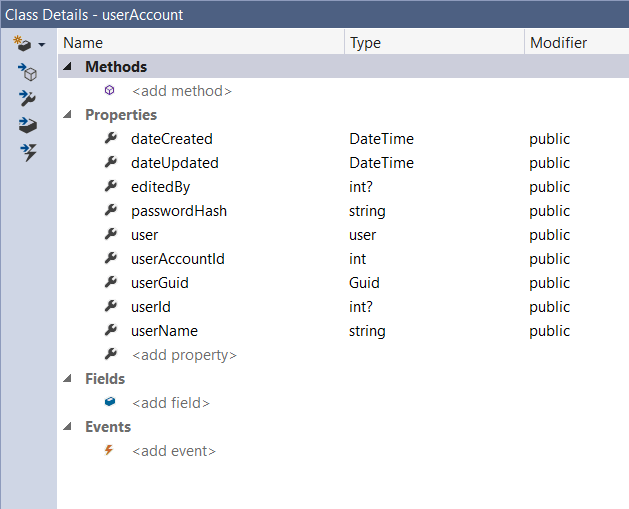
**User**

The user class contains the key data properties for registered system users.

****

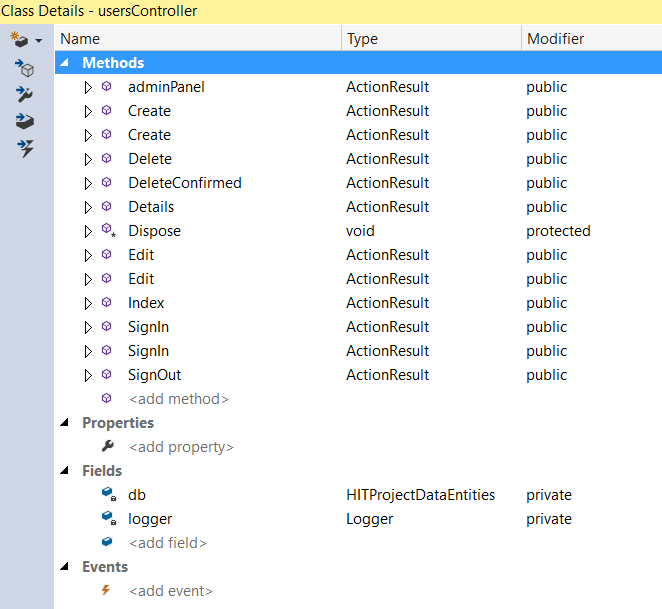
**User Account**

**The user account class contains the properties related to user accounts for registered system users.**

****

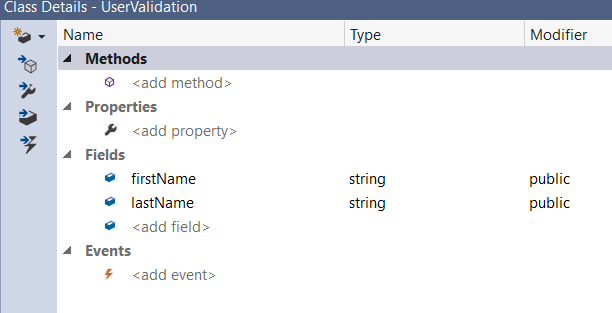
**Users Controller**

The users controller contains the expected controller methods for users.

****

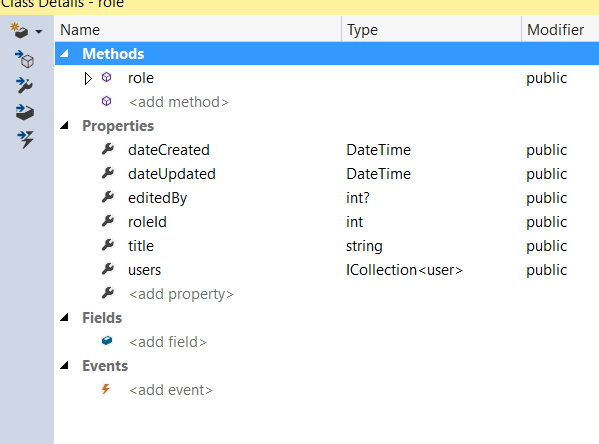
**User Validation**

The user validation class contains the fields required for user validation according to current application configuration.

****

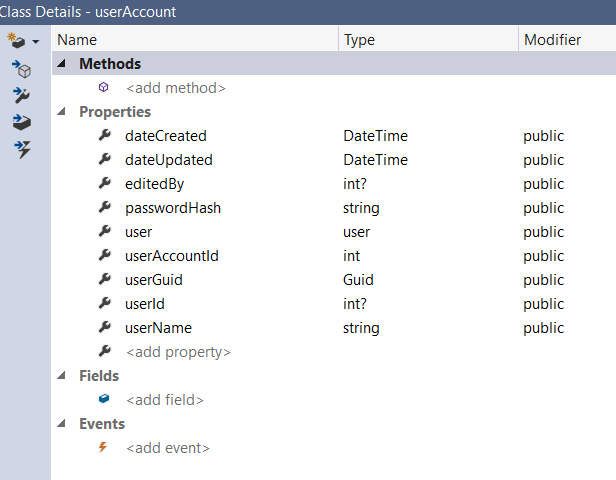
**Role**

The role class contains the methods and properties required for user roles.

****

**User Account**

The user account class contain the current properties configured for user accounts.

****