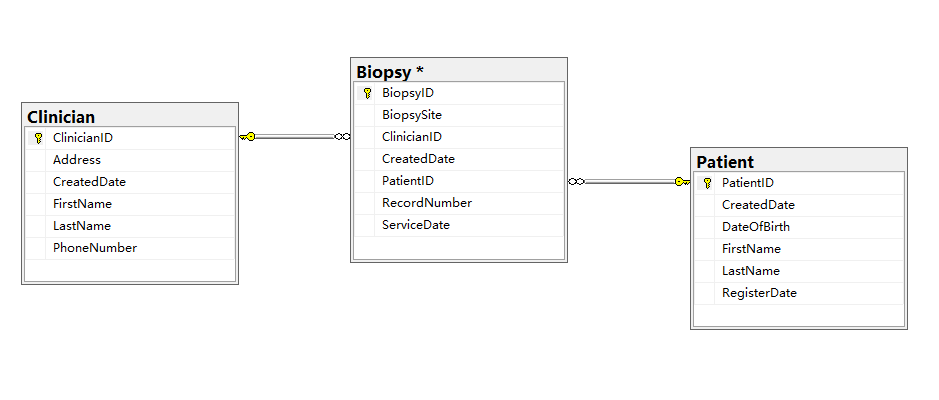
JHU Patient Biopsy System

1. Overview
2. Database
   1. ER Diagram
   2. Patient Table
   3. Clinician Table
   4. Biopsy Table
3. System Detail
   1. Models
   2. Views
   3. Controllers
   4. System snapshots
4. **Overview**

The system is implemented using C# ASP.net Core 2.0 MVC and Entity Framework 6.0. The developing environment of this project is Visual Studio 2017. The database for this project is MS SQL Server with a localDB instance.

The objective of the system is to provide a web application that can display and manage the patient and biopsy information, which are stored in a MS SQL Server.

1. **Database**
   1. ER Diagram



In this demo system, the database includes only three tables: Patient, Biopsy, and Clinician. The corresponding ER diagram is shown above. The Patient table has a 1-to-n relationship with the Biopsy Table, indicating that for each patient there could be a set of biopsy visits. Similarly, there is a 1-to-n relationship connection from the Clinician table to the Biopsy table. The detail information about each table will be described in the following sections.

* 1. Patient Table

**Field List:**

PatientID – Automatic generated key field, int

FirstName – The first name of patient, nvarchar(MAX)

LastName – The last name of patient, nvarchar(MAX)

DateOfBirth – The birth date, in a format of YYYY-MM-DD, datetime

RegisterDate – The date when the patient is registered in the system, datetime

CreateDate – The date that the record is created, datetime

**Description:**

The Patient table includes the general information about each patient. The key column of the Patient Table is the PatientID, which is a database generated field. The DateOfBirth, FirstName, LastName and RegisterDate columns store the profile information about the patient, and the CreatedDate column indicating the create date of the record. In this project, the created date is inputted by users manually, while in a real system, the CreatedDate column could be filled automatically by the system when the record is being created.

**Possible extension to the table**:

additional attributes of the patient, such as race, gender, could be included if needed; additional information regarding to the operation to the record, such as an UpdatedDate column, could be included to indicating the latest modification time of the record; if the system has authentication, and has a table managing the users of the system, possible foreign key fields indicating the creator/updater could be included, to store the information about who create the record, and which user made the latest modification to the record.

* 1. Clinician Table

**Field List:**

ClinicianID – Generated key field, int

FirstName – The first name of clinician, nvarchar(MAX)

LastName – The last name of clinician, nvarchar(MAX)

Address – The address of the clinician, nvarchar(MAX)

Phone – Phone number, nvarchar(MAX)

CreatedDate – The date that the record is created, datetime

**Description:**

In this system, the clinician information is separated from the biopsy table, and stored and managed in the Clinician table. The reason of this design is to provide an extendable way for managing the clinicians’ information, and to prevent inconsistency when some of the information is changed.

Similar to the patient table, the general information is stored in this table, and a database generated ClinicianID field is included as the key column.

**Possible extension:**

Similar to the patient table, additional clinician information, CreatedBy, UpdatedBy, UpdatedDate columns could be included if needed.

* 1. Biopsy Table

**Field list:**

BiopsyID – Generated key field, int

RecordNumber – The assigned record number, UNIQUE, nvarchar(20)

BiopsySite – The site name of the biopsy, nvarchar(MAX)

ServiceDate – The date of the visit, datetime

ClinicianID – Foreign key field indicating the clinician of the biopsy, int

PatientID– Foreign key field indicating the patient of the biopsy, int

CreatedDate – The date that the record is created, datetime

**Description:**

The biopsy table stores the related information for each biopsy visit. In this demo, the biopsy table has two foreign key fields, the ClinicianID and the PatientID, which can connect each biopsy record to the corresponding patient record and clinician record. The Record Number indicate the associated record number. Although the record number can be treated as a key field, a database generated biopsyID field is included as the key field in this demo. The major reason for this design is that the record number is a user defined indicator, and the definition could be changed in the future, so by introducing an individual key field, the database could be more flexible to such modification.

**Possible extension:**

In this demo, the biopsy site is assumed to be any string value indicating any part of human body, and thus with no restriction to the input. In a more restricted design, a reference table, BiopsySite, can be introduced into the database. The BiopsySite table stores a list of possible site name, and associate a reference id to each site. Then the biopsy site field in Biopsy table can be changed to a foreign key pointing to the BiopsySite table.

1. **System Detail**
   1. Models

In this demo, 3 models (Patient, Biopsy, Clinician) are created in corresponding to the three tables in the database.

In Patient and Clinician model, a generated class member “FullName” is included to represent the full name of the patient/clinician, the value is generated by concatenating the first name and the last name.

All the Models files are located in the /Models folder.

* 1. Views

In this section, a list of views and its objective is described.

* + 1. Biopsies Views

Index – Listing all the biopsies in the system

BiopsyList – Listing all the biopsies with a given patient ID

Create – Creating a new biopsy, no patient ID required

CreateWithID – Creating a new biopsy, with a given patient ID

Delete – Displaying and deleting a given biopsy record

Edit – Editing a given biopsy record

Detail – Showing the selected biopsy record

* + 1. Patients Views

Index – Listing all the patients in the system

Detail – Showing the detail of the patients, including the biopsies

Create – Creating a new patient record

Edit – Editing the patient information

Delete – Deleting the patient and his/her biopsies

* + 1. Clinicians Views

Index – Listing all the clinicians in the system

Detail – Showing the detail of the clinician

Create – Creating a new clinician record

Edit – Editing the clinician information

Delete – Deleting the clinician record

* + 1. Home Views

About – Showing the about page of the system

Contacts – Showing my contact info

Index – Showing the home page of the system

* 1. Controllers

In this section, a list of function interfaces for each controller is listed, together with their binding URL and HTTP method (GET or POST).

* + 1. Biopsies Controller

Index() – HTTP GET: Biopsies/

Retrieving all the biopsies, and send the data to the view page: Biopsies/Index.cshtml;

BiopsyList(int?) – HTTP GET: Biopsies/BiopsyList?patientID=1.

Retrieve the biopsies with the given patientID, send the data to view page: Biopsies/BiopsyList.cshtml. Throw an exception if the patientID is not valid

Details(int?) – HTTP GET: Biopsies/Details/1.

Retrieve the biopsy record with the given id, send the data to view: Biopsies/details.cshtml;

Create() – HTTP GET: Biopsies/Create

Prepare the selection lists for the creating view, direct to view: Biopsies/create.cshtml;

CreateWithID(int?) –HTTP GET: Biopsies/CreateWithID?patientID=1

Retrieve the patient name with given PatientID, throw an exception if the given patient ID is not found in the system; return to view: Biopsy/CreateWithID.cshtml.

Create(Biopsy) – HTTP POST: Biopsies/Create

Create the record with the given posted biopsy instance. Return redirect to the Index page after creation.

CreateWithID(Biopsy) – HTTP POST: Biopsies/CreateWithID

Create the record with the given posted biopsy instance. Return redirect to the BiopsyList page after creation.

Edit(int?) – HTTP GET: Biopsies/Edit/1

Retrieve the biopsy record with the given BiopsyID, and prepare for the editing view: Biopsies/Edit

Edit(int, Biopsy) – HTTP POST: Biopsies/Edit/1

Update the database with the posted biopsy instance. Return redirect to Biopsies/BiopsyList page after edition.

Delete(int?) – HTTP GET: Biopsies/Delete/1

Retrieve the biopsy record to be deleted, and direct to view: Biopsies/Delete.cshtml

DeleteConfirmed(int?) – HTTP POST: Biopsies/Delete/1

Delete the biopsy record from the database, and return redirect to Biopsies/BiopsyList view

* + 1. Patients Controller

Index() – HTTP GET: Patients/

Retrieving all the Patients, and send the data to the view page: Patients/Index.cshtml;

Details(int?) – HTTP GET: Patients/Details/1.

Retrieve the Patient record with the given id, then retrieve the biopsy records belong to the patients. Send the data to view: Patients/details.cshtml;

Create() – HTTP GET: Patients/Create

Direct to view: Patients/create.cshtml;

Create(Patient) – HTTP POST: Patients/Create

Create the record with the given posted Patient instance. Return redirect to the Index page after creation.

Edit(int?) – HTTP GET: Patients/Edit/1

Retrieve the Patient record with the given PatientID, and prepare for the editing view: Patients/Edit

Edit(int, Patient) – HTTP POST: Patients/Edit/1

Update the database with the posted Patient instance. Return redirect to the Index view after edition.

Delete(int?) – HTTP GET: Patients/Delete/1

Retrieve the Patient record to be deleted, and direct to view: Patients/Delete.cshtml

DeleteConfirmed(int?) – HTTP POST: Patients/Delete/1

Delete the Patient record and all the biopsy records belong to the patient from the database, and return redirect to the Index view

* + 1. Clinicians Controller

Index() – HTTP GET: Clinicians/

Retrieving all the Clinicians’ records, and send the data to the view page: Clinicians/Index.cshtml;

Details(int?) – HTTP GET: Clinicians/Details/1.

Retrieve the Clinician record with the given id. Send the data to view: Clinicians/details.cshtml;

Create() – HTTP GET: Clinicians/Create

Direct to view: Clinicians/create.cshtml;

Create(Clinician) – HTTP POST: Clinicians/Create

Create the record with the given posted Clinician instance. Return redirect to the Index page after creation.

Edit(int?) – HTTP GET: Clinicians/Edit/1

Retrieve the Clinician record with the given ClinicianID, and prepare for the editing view: Clinicians/Edit

Edit(int, Clinician) – HTTP POST: Clinicians/Edit/1

Update the database with the posted Clinician instance. Return redirect to the Index view after edition.

Delete(int?) – HTTP GET: Clinicians/Delete/1

Retrieve the Clinician record to be deleted, and direct to view: Clinicians/Delete.cshtml

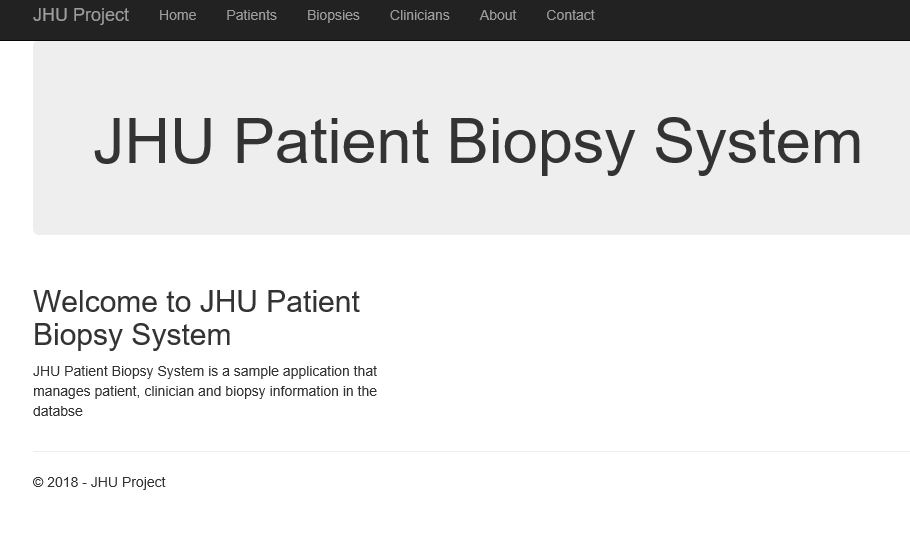
DeleteConfirmed(int?) – HTTP POST: Clinicians/Delete/1

Check if the clinician record is referred by any biopsy, if the clinician record is being used, return an error message; otherwise, delete the clinician record from the database, and redirect to the Clinicians/Index view.

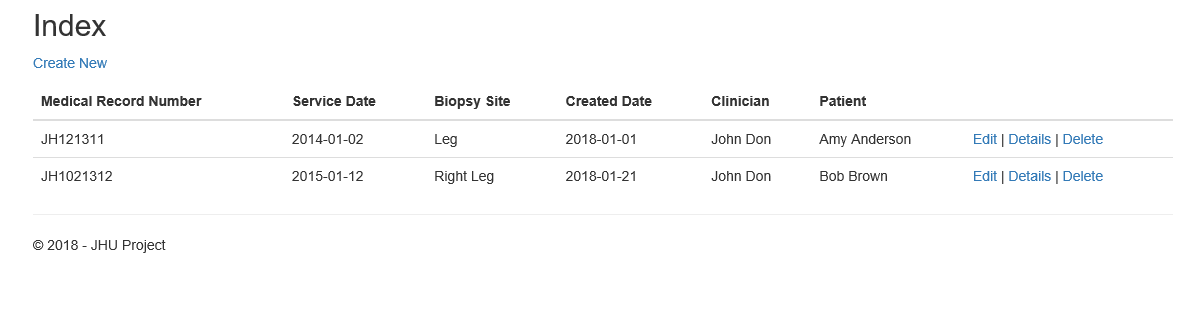
* 1. System Snapshots

All the snapshots can be found in the archived package

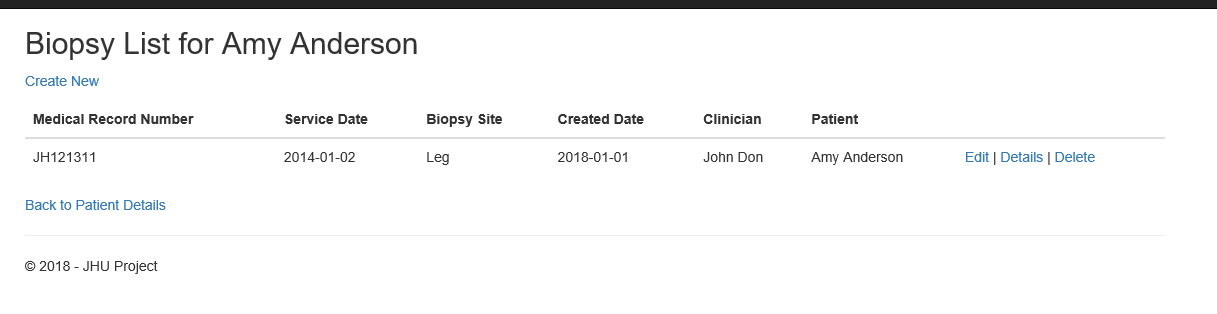
Home page:



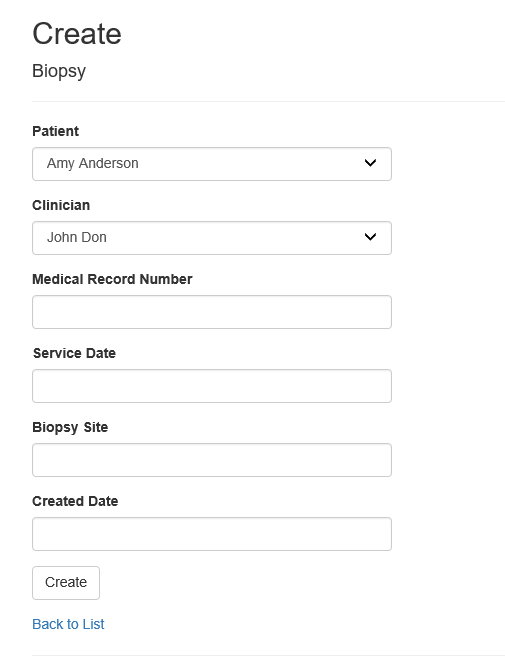
Biopsy Index page:



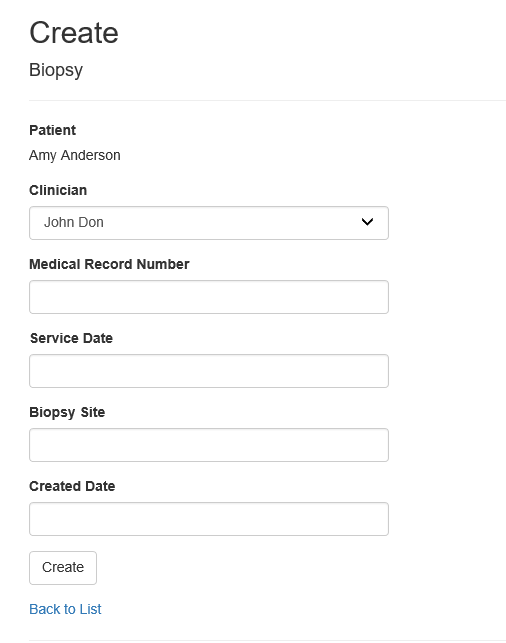
BiopsyList with given PatientID:



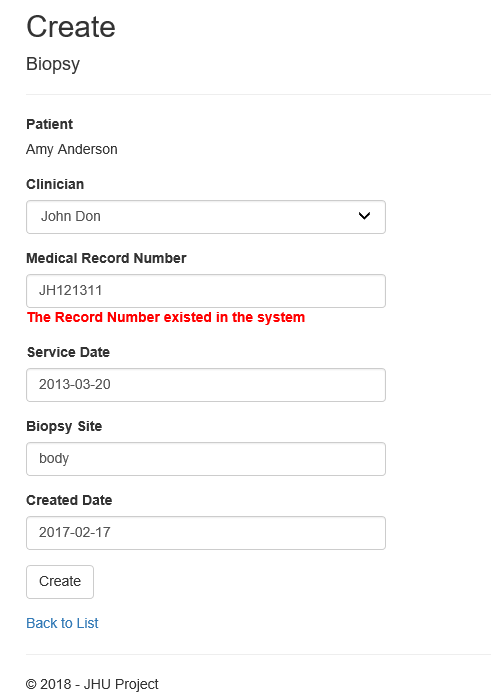
Biopsy Create page:



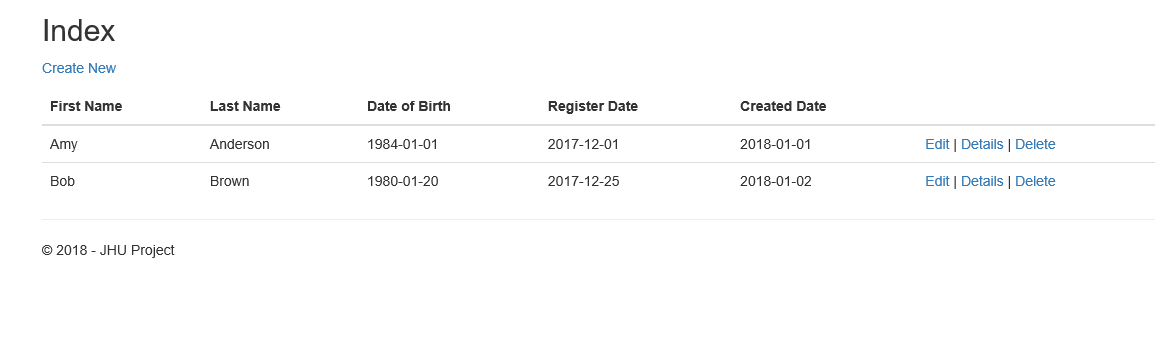
Biopsy Create page with PatientID:



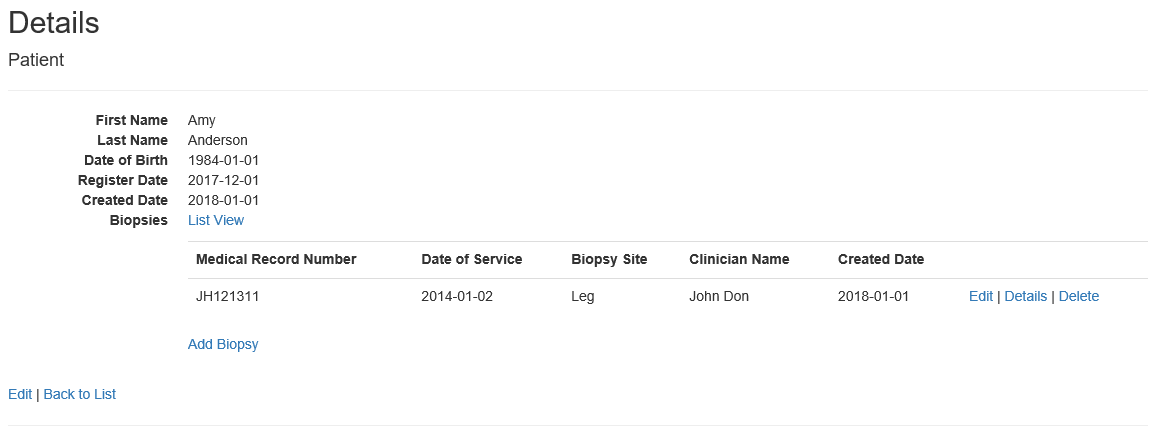
Biopsy Create, validate the Record Number:



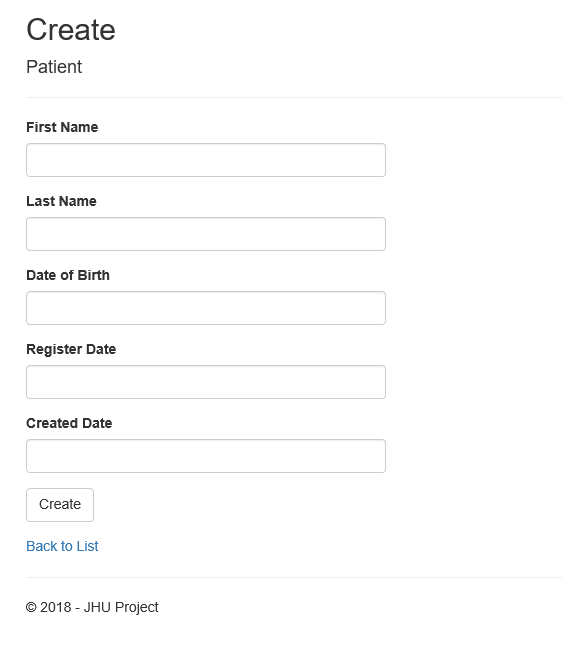
Patient List:



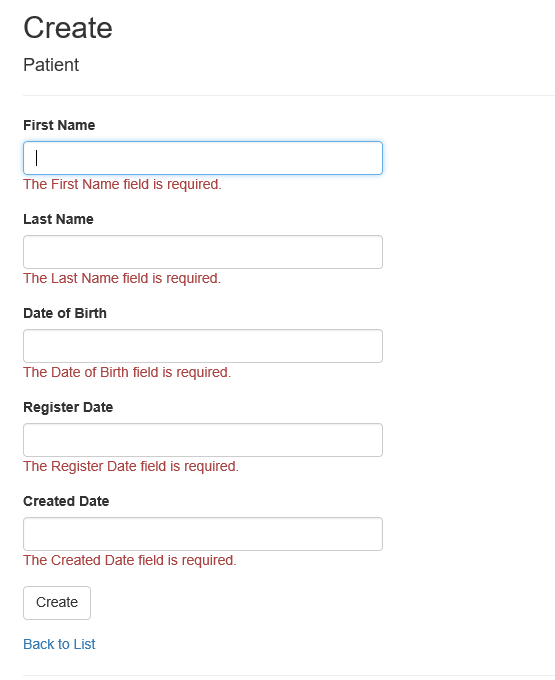
Patient Detail page, with biopsies



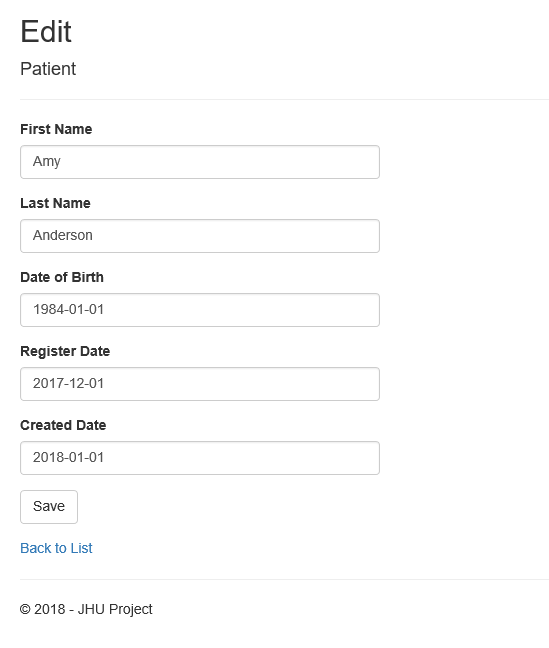
Patient Create page:



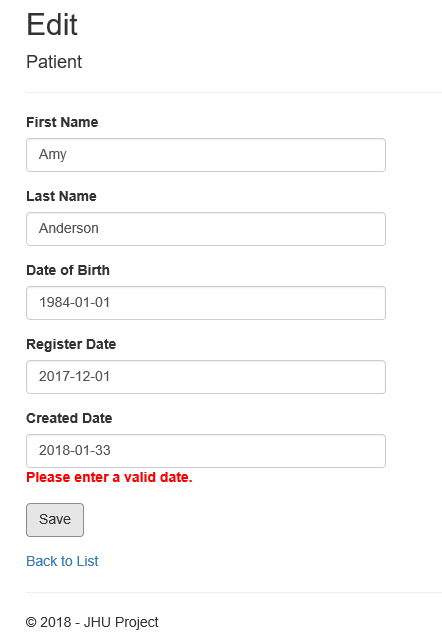
Patient Create Page: Validating the data



Patient Edit page:



Patient Edit, validation:



Clinician Delete, validation:

