This is my rough understanding of the Flex structure used in the MC project.

Directories and Files that are relevant:

All Flex files and directories are under */app/flex*.

The main mxml file is located at */app/flex/Matlabulchifai.mxml*. It displays the main page and uses a set of components that are located at */app/flex/matlabulchifai/components/*. Each component defines an event handler that handles events that are triggered by elements of the component. Event classes are defined in */app/flex/matlabulchifai/events/*.

The Flex general structure:

In general the mxml files encountered in this application can be divided in 4 parts.

1. **The header:** where general/standard page layout and settings are defined. The header is in general not relevant to us.
2. **The script:** where action script methods and functions are defined. These are the methods that take care of the user interactions with the application. They handle events, interact with the database and redirect request results to the user. It corresponds in a large extend to the controller in Rails.
3. **The services (remote procedure calls RPC):** these services are called by the event handlers when there is need to communicate with the database. Two main services are used in this project.
   a. **Httpservice:** it is one of the most common components used to retrieve data in Flex. The most common httpservice calls seen in this project look like:
      ………………………………………………………………………………………………
      ```
      <mx:HTTPService
       id="svcAccountLogin"
      url="/session.xml?authenticity_token={Application.application.parameters.authenticityToken}"
      resultFormat="e4x"
      method="POST"
      result="handleAccountLoginResult(event)"/>
      ```
      ………………………………………………………………………………………………
      The *id* is the unique name given to this service in this file. It is used by the event handlers to make calls to the correct service. The *url* is the exact address where the data should be accessed. The *resultFormat* defines how the results of a given request are presented to the caller. The *method* (POST or GET) is the actual httpservice method called when the action is called. *Result* is the event handler that is called once the result of a given request is available. The handler should be a method/function implemented most of the time in the script part.

b. **XMLListCollection:** *I should study this one more ☺*, but basically it is an array of objects returned as a list. It provides a set of functionalities and methods to perform certain xml tasks.

4. **The page layout:** in this part, are defined the different components of the view. This is literally the view part of an MVC model.
   Components in Flex mxml have a property called *event*. This allows to the developer to define functions that are called when the given event happens. Those functions should be defined in the script part of the file.

The matlabulchifai.mxml file:

Below is shown the different parts of the MC main mxml file.

Title and Language bar in an Hbox component (these are fixed elements that do not change during the entire user experience)
Menu bar: in user or admin mode. This is set to *invisible* before login time and then set to visible for the rest of user experience
Main stack: the dynamic part of the page that has several children (loginBox, alertGridBox,patientBox, searchBox, adminBox, patientInfoBox). The loginBox is loaded when the user first points to the application. After successful login, the alertBox is loaded and depending on user actions other components (children) are loaded.

At *init* time, a bunch of event listeners are defined that listen for the different actions that a user might take. Once an event occurs, the corresponding event listener is triggered and calls the corresponding event handling. The event handler analyses the event and if needed calls the appropriate service to access the model. If a service is called, it eventually returns a result that is handled by a result handler method. Once the result event is taken care of, the result is displayed to the user by loading the right child to the main stack.
As an example, here is how the loginBox works.

The loginBox is defined as a component of the main application in the directory */app/flex/matlabulchifai/components/LoginBox.mxml* and has an event listener class that is a subclass of the predefined class Event and is defined in */app/flex/matlabulchifai/events/login/LoginEvent.as* (all components that require some event listener must implement an event class that extends Event).
It contains a *Form* that has 3 form items: two text inputs (username and password) and a button (the login button) that are bound together.
On click (a property of a button) event, the login button calls the event handler *login()* (which is a function defined in the script part of the LoginBox.mxml file). The *login()* function calls the *POST* method of the httpService via the service id *svcAccountLogin*. This is done by invoking the *send* method of the

service via the service id and with the parameters (username, password). The result of this invocation is handled by the function *handleAccountLoginResult()* which is also defined in the script section of the file. If the login result is OK, the user (which is retrieved from the model via the url "/session.xml?authenticity_token={Application.application.parameters.authenticityToken}") is sent to the main page via the method *dispatchEvent().*

On the main page, login event is handled by the method *handleLogin()* . It just calls the method *login()* (in this file not the one in *LoginBox.mxml*)with the returned user as parameter. This method set the visibility of the menu bar to TRUE and shows it after determining whether the user has administrator privileges or not. It also checks whether the language is English, French, or Hindi and accordingly set the view. Then, it calls the method *listPatients()* to list the patients. The *listPatient()* method retrieves the list of the patients by invoking the httpService via the service id *svcPatientList.* The result of this request is handled by the method *handlePatientListResult().* This later method stores all the returned patients in the variable *patientIdMap* and by calling the method *updatePatientIdMap(),* then it tests whether this is the first time that the patient list s loaded (via the global variable *firstPatientLoad*). If it is so, it set the *alertBox* as the current child of the main stack and list patient by their alerts.

The same logic is used whenever an action is triggered by the users.

Action Items:
To move forward in our Flex-Html migration, we should implement both the views and the controllers for the application. For the views, we should refer to the *matlabulchifai.mxml* file and to the different components that are in */app/flex/matlabulchifai/components/* mostly into the layout part of the file. In fact; both the view layout and the different components of the view are available in those files. The controllers should also be implemented by referring to the same files but mostly in the script part. The logic of the controllers can be understood by observing those files.