CSCI4448: Project Part 2

Alexia Newgord, Mark Lohaus

Project Summary

Snap for Life is a web application that enables emergency medical responders to collect immediate patient information and images in the field, such as at the scene of an accident or in an ambulance. They will be able to select the type of accident and a receive a simple set of customizable questions to answer about the patient's condition. The set of questions will be configurable ahead of time by hospital administration to match a particular medical situation. The medical responders on site will be able to upload the photos and information of the patent's energies to be accessed quickly and securely via a web page by the emergency room personnel.

The objective of this application is to reduce hospital mortality rates by improving the speed and quality of communication between medical emergency responders and hospitals in order to enable better preparation and care for the patient. The data will not be stored on the user's personal mobile device so that HIPAA violations are safely avoided.

Project Requirements

Requirements and Associated Responsibilities:

Req. #1: Emergency responder may submit patient information

- 1.1 Emergency responder logs into web application
  - 1.1.aSystem authenticates user. *Stretch* - Web application remembers mobile device, so that responder only needs to input PIN to quickly access application
- 1.2 Emergency responder takes image(s) with mobile device
- 1.3 Emergency responder checks applicable fields in mobile application form
- 1.4 Emergency responder adds comments in mobile application form
- 1.5 System should support Android and iOS devices
  - 1.5.aSupport differences in screen size, camera functionality, mobile browser
- 1.6 System should support a minimum of 15 responders at a time
- 1.7 Keep track of which users are logged in (*Stretch* - Hospital can see which responders are logged in and their current location).
  - 1.7.a Have the system/database remember which responder logged the submission
  - 1.7.b There should only be one responder logging information for a patient. This should be expressed to the responders, but system may throw warnings if it suspects duplicate entries. *Stretch* - Designated logger can pass of their responsibility to another logged-in responder if they have to leave the scene.

Req. #2: Hospital may view submitted patient information

- 2.1 Hospital logs in via website
  - 2.1.a System authenticates user
- 2.2 Hospital receives reports/images via website

2.3 Hospital prioritizes patients in queue and may mark patients as “treated”

2.3.a When patients are marked as “treated,” system confirms this is the desired action and removes them from the main view

2.4 System supports any mainstream browser and operating system

2.4.a IE, Firefox, Safari, Chrome

2.5 System supports a single hospital user at a time (*Stretch* - Only one hospital user can view *and* modify queue, but additional authenticated users may simultaneously view patient information).

Req. #3: Hospital Admin may customize users and questions

3.1 Admin logs in via website or mobile application

3.1.a System authenticates user

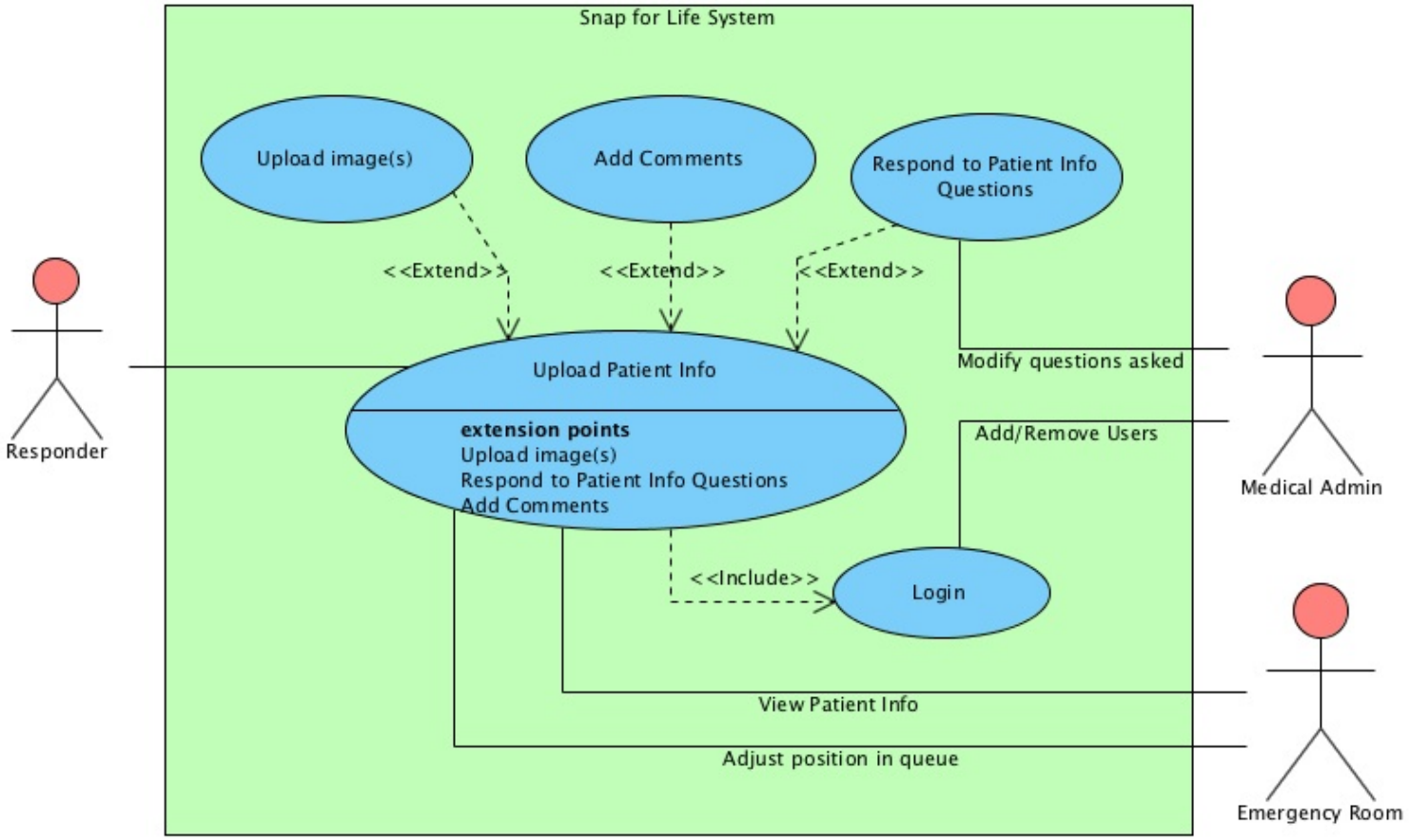
3.2 Admin adds/removes users via backend database, determines authorization based on hierarchical structure

3.3 Admin customizes questions that are asked to responders

3.4 System should support any mainstream browser and operating system

Users and Tasks

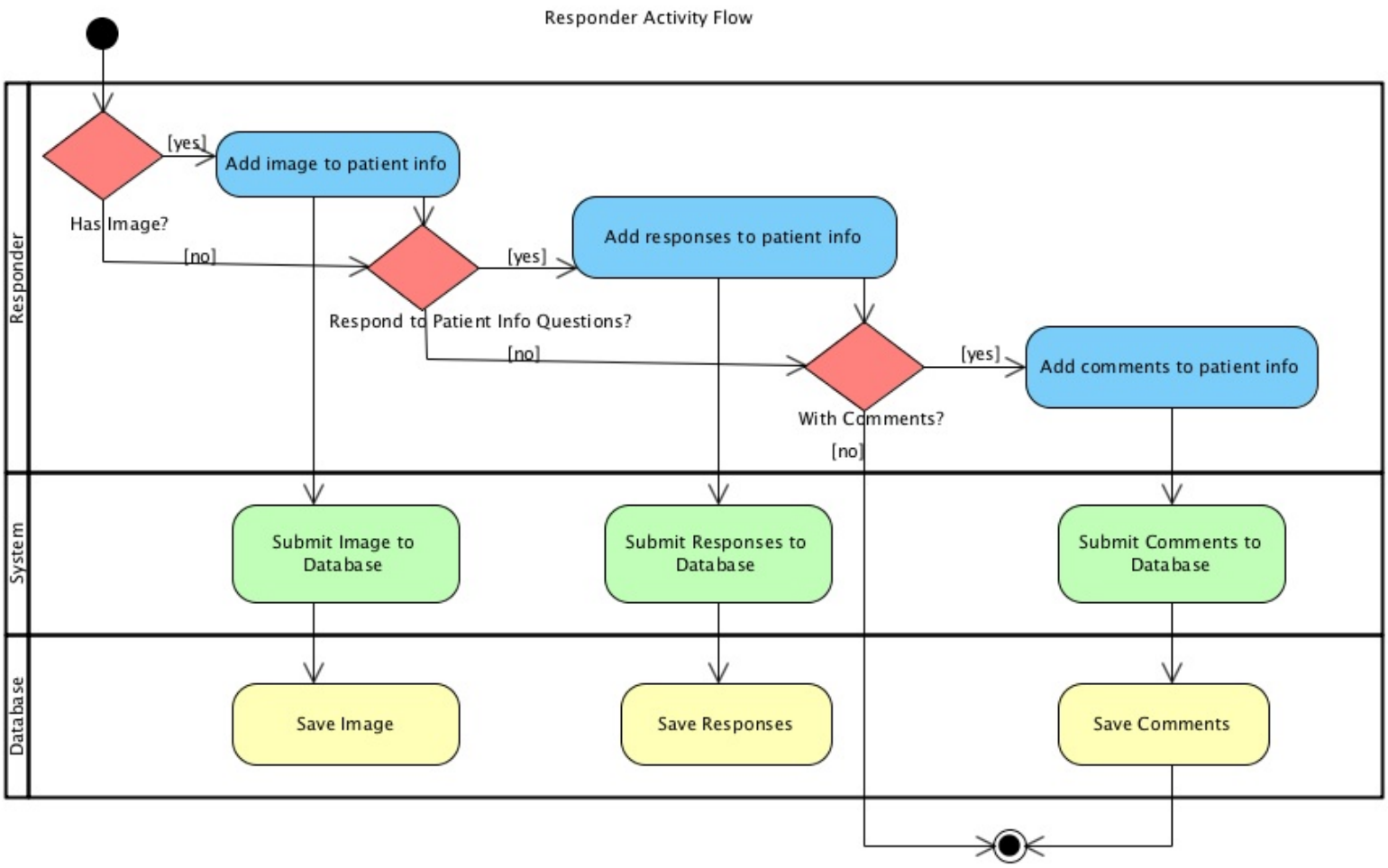
The system has three different types of users: Responders, Emergency Room, and Medical Admin. The tasks that they can accomplish with the system are outlined in the use case diagram below:



Use Case ID:	UC-01	
Use Case Name:	Upload Patient Info	
Description:	Emergency Responder may use web application to instantly provide images and information to the hospital about patient on the scene.	
Actors:	Emergency Responder	
Preconditions:	- The responder must have a working iOS or Android mobile device with internet access - Questions for the responder must be configured beforehand by administrators	
Postconditions:	on success:	Images are submitted through the system to the database, user is notified of receipt
	on failure:	User is notified of error and is given phone number of hospital for a backup means of communication
Trigger:	User opens web application on mobile device	
Basic Flows:	1. User logs in and system verifies user access 2. User uploads image (optional), received by system 3. User responds to questions from the app (optional), received by system 4. User adds comments regarding situation (optional), received by system 5. User logs out or is automatically logged out by system	
Alternative Flows:	- User login is rejected by system - User has the option to return to the questionnaire, comments, or image section to modify or update information	
Notes and Issues:	Communication with the web application may be hindered by poor cellular reception	

Use Case ID:	UC-02	
Use Case Name:	View Patient Info	
Description:	Emergency Room may view the information submitted by the emergency responders	
Actors:	Emergency Room	
Preconditions:	There must be patient information loaded in the database	
Postconditions:	If exists:	Patient information is displayed
	If doesn't exist:	User is notified that there is no current patient information
Trigger:	Requesting access to patient information	
Basic Flows:	1. User logs in and system verifies user access 2. User requests access to patient information 3. System displays current patient information 4. User logs out	
Alternative Flows:	User login is rejected by system	
Notes and Issues:	By default, system does not display information for patients that have already been tended to	

Activity Diagram



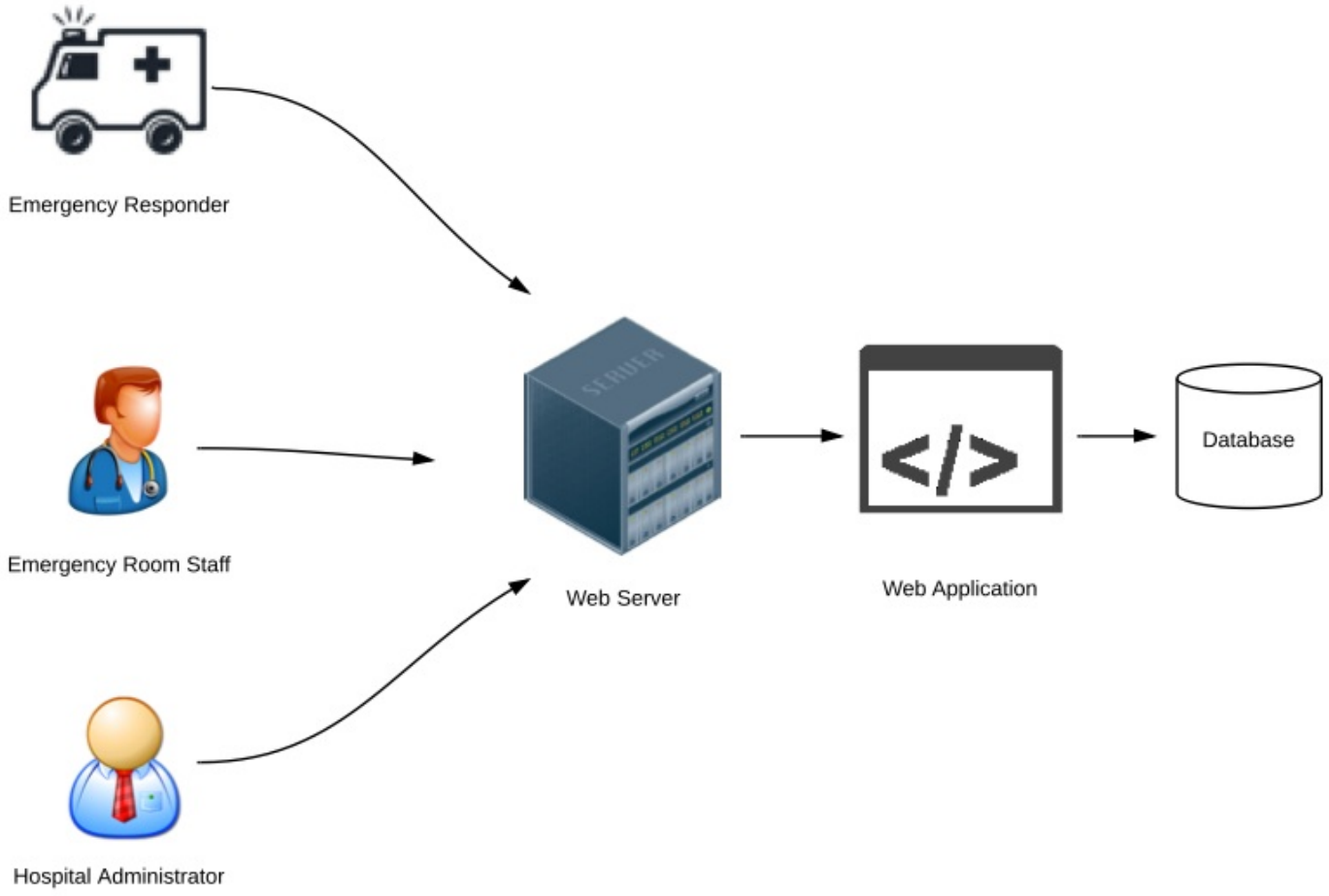
Architecture Diagram

The bulk of the system is a web application that has a view for desktop computers and a view for mobile applications. We will be using Yii framework to implement this in an MVC format. Yii appears to be the ideal choice for this application because it is considered secure and its use of Lazy Loading techniques increases query speeds (which is critical for such a time-sensitive application). It is also open source, used with PHP, and has a handy debug mode.

Some Yii features that will be implemented include:

- CHtml for development
- Form-validation functions for logging in
- User logging
- Database Access Objects
- Layouts - Themes, skins

All users will be participating in request-response cycles with the web application. For example, when an emergency responder marks “TRUE” for a pre-configured question regarding the state of the patient, an HTTP request will be submitted from the user’s browser. After the request is handled by the Yii framework controller and model, the HTTP response will be sent back to the user’s browser.



Data Storage

The model classes will communicate with a relational database and transform the relational data into an object-oriented representation of that data. An ORM (object relational mapping) tool, such as Doctrine, will be implemented. Hand written SQL encapsulated in discrete model methods may also be used to conduct CRUD (create, read, update and delete) database operations.

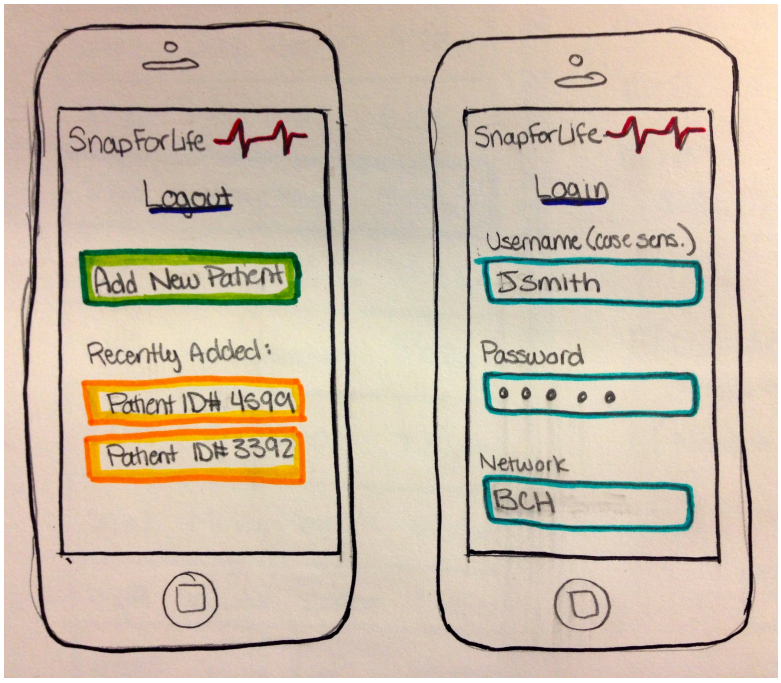
The Yii Database Access Objects (DAO) also provides a generic API to be used as an additional layer of abstraction for accessing different database management systems (DBMS), such as MySQL or PostgreSQL. The DAO primarily consists of the CDbConnection, CDbCommand, CDbDataReader, and CDbTransaction classes.

The primary key for patient information is the Patient ID, which is uniquely assigned by the system to keep the record distinct from others and allow all other patient information fields to be optional.

UI Mockups

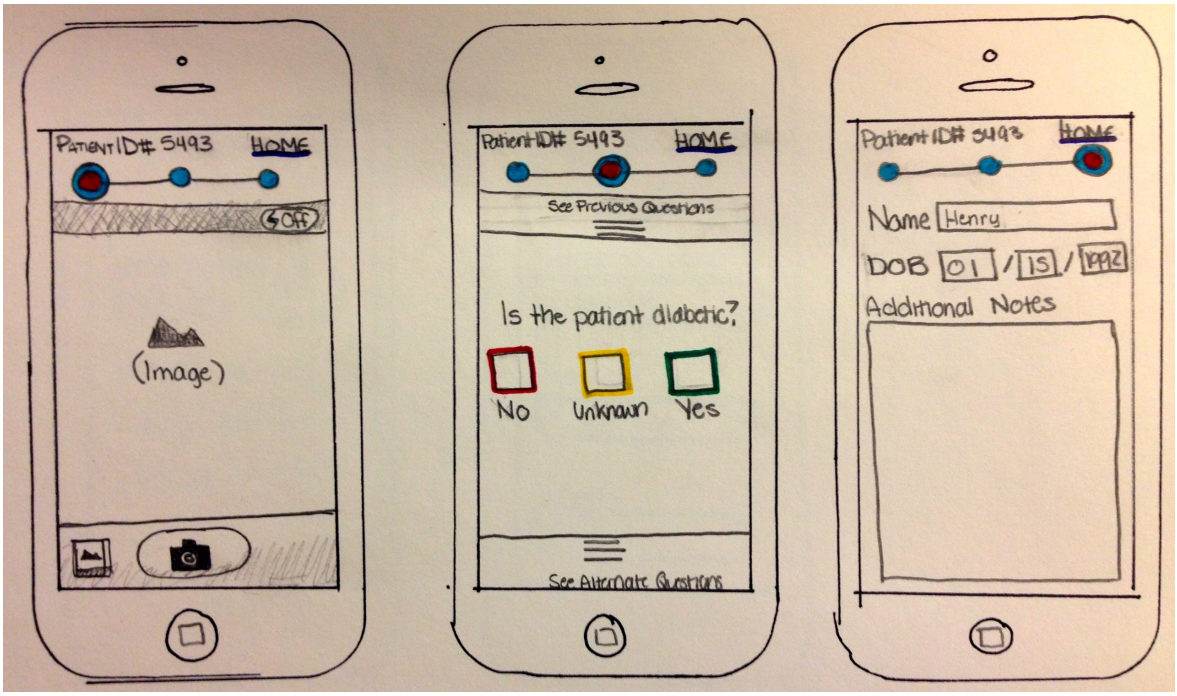
Emergency Responder Home Pages (on iPhone5 iOS device):

Responder may tap on the Recently Added “Patient ID#...” to modify earlier submissions or tap “Add New Patient” to create a new patient information submission. They may also logout (if logged in) or login (if logout) given the correct authentication information.



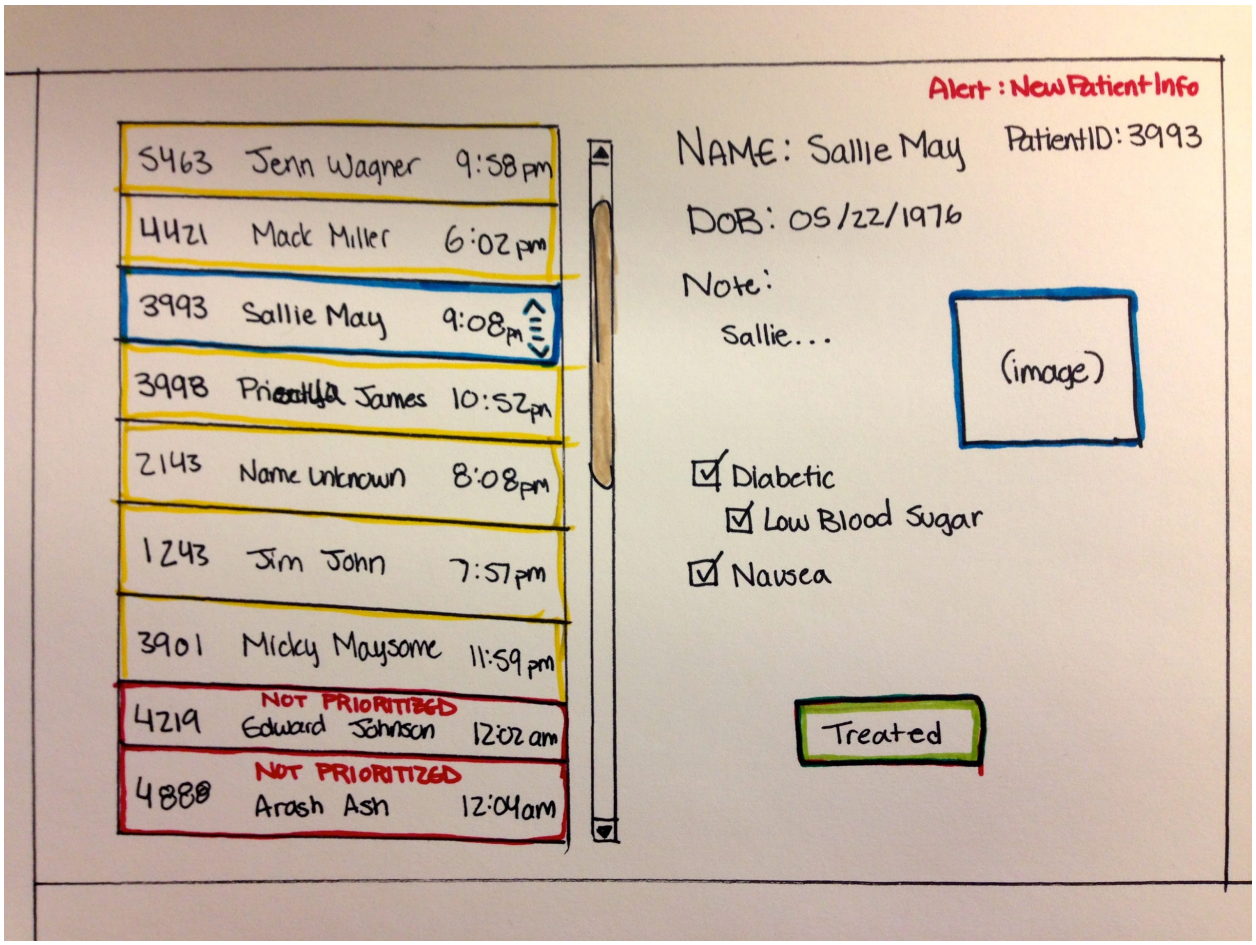
Emergency Responder Patient Info Pages (on iPhone5 iOS device):

Tapping on “Add New Patient” or “Patient ID#...” will bring the responder to the following three screens, which are navigated by tapping a circle on the top blue bar. The first view takes an image. The second view asks pre-configured questions based on the prior answers. If the user wishes to modify earlier answers, he/she can pull down the “See Previous Questions” menu. If they feel like they are answering the wrong family of questions, they can pull up the “See Alternate Questions” menu. Finally, they can input general comments an information in the third view or return home by tapping “Home” at any time.



Emergency Room User Interface (on standard browser):

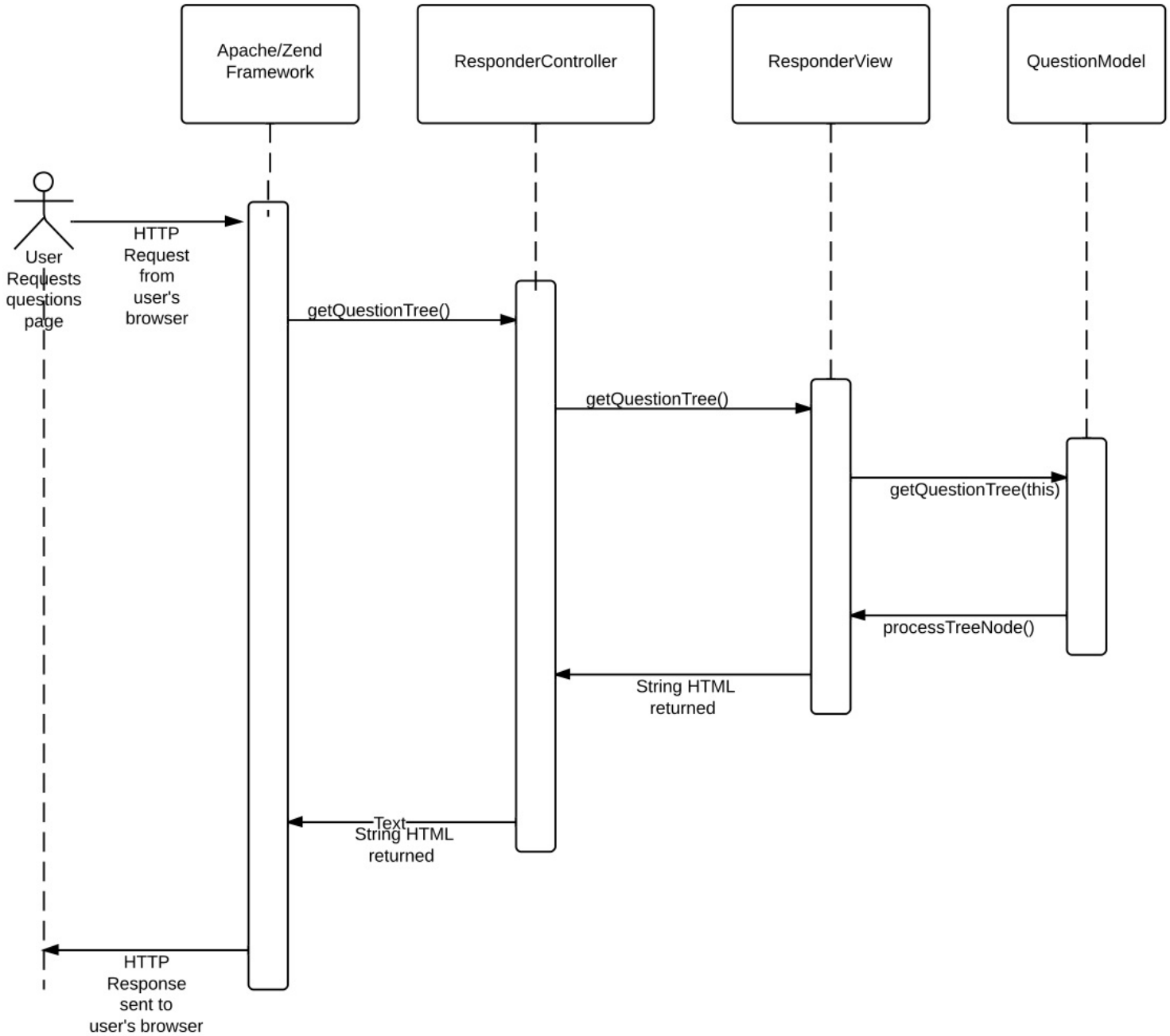
After logging in the emergency room personnel may view the submitted patient information. A new submission is placed at the bottom of the list (with the idea that the top priority patients are listed from top to bottom), but is marked red to alert viewers that it has not been viewed or prioritized. There is also an alert in the top right corner just in case the submission is out of view. When the user clicks on a submission, the color turns blue and the details of the submission is displayed on the right. Once selected, the user may optionally re-prioritize it within the queue or click “Treated” to have it removed from the view. (Note: the prioritizing feature of the queue may be a stretch functionality)



User Interactions

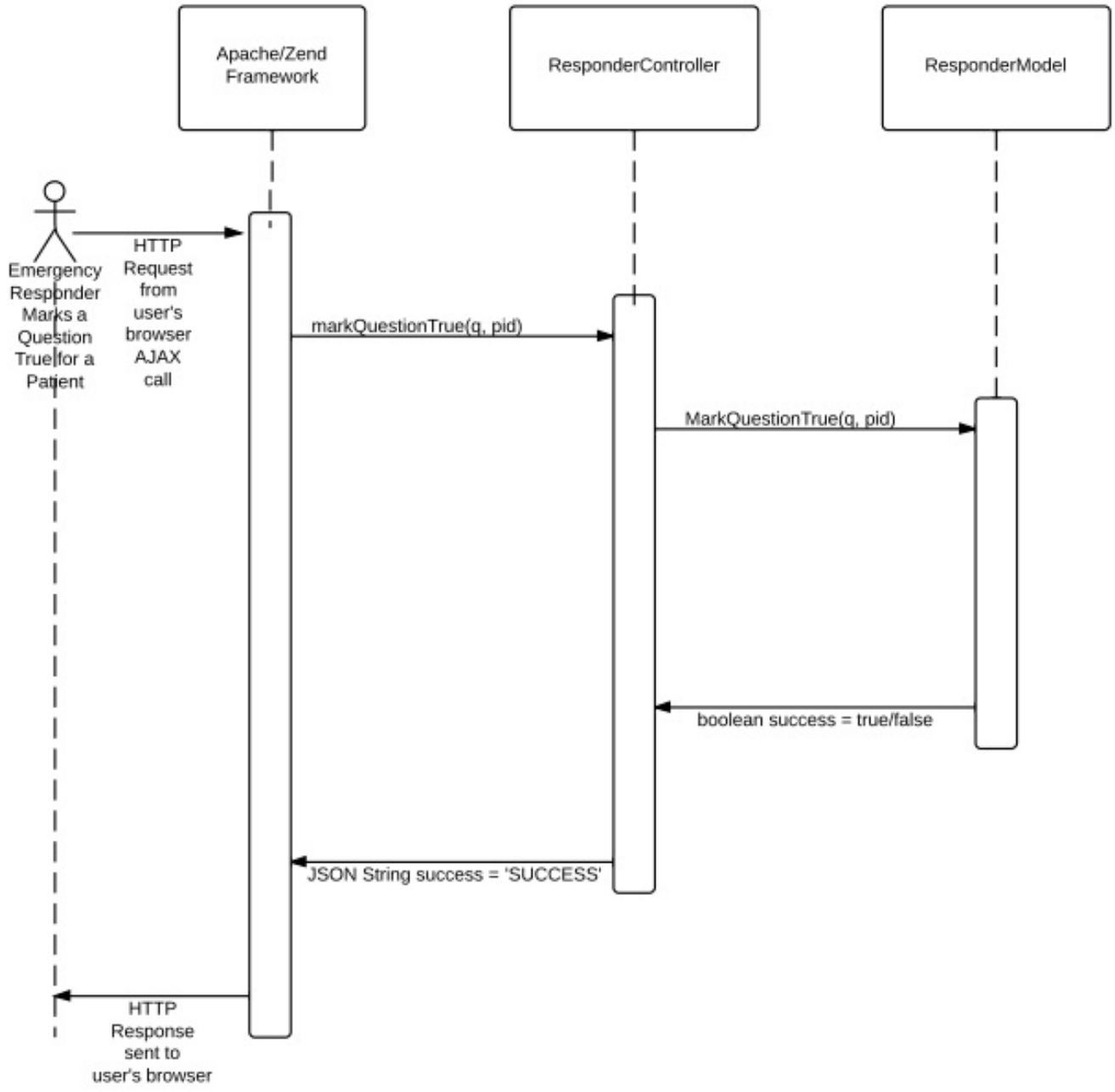
Interaction #1: The sequence diagram below describes the process that occurs when the emergency responder elects to answer questions about the patient's condition. When this happens, the framework will receive the HTTP request from the user's browser. The framework will communicate with the ResponderController, which passes on the request to the ResponderView, which finally retrieves the question tree from the QuestionModel. The ResponderView returns the response to the ResponderController, the Framework, and the user's browser in that respective order.

Emergency Responder Requests Questions Page sequence diagram



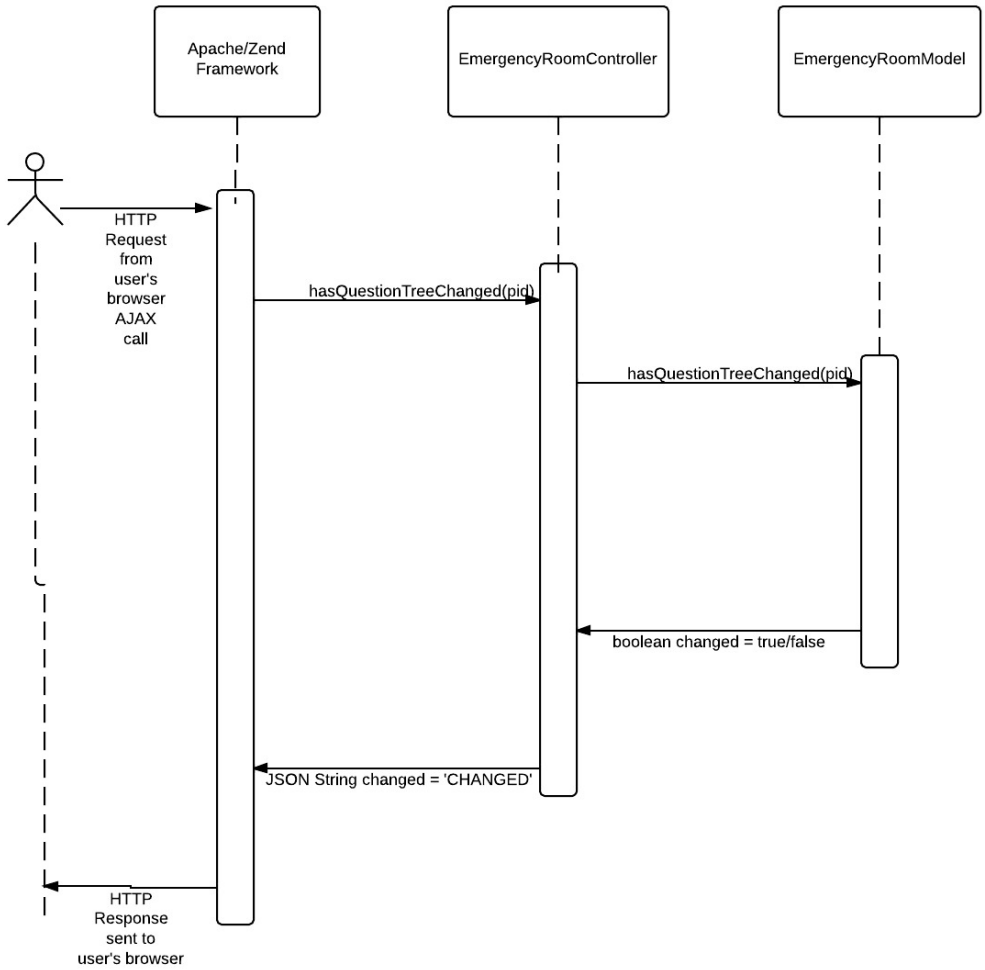
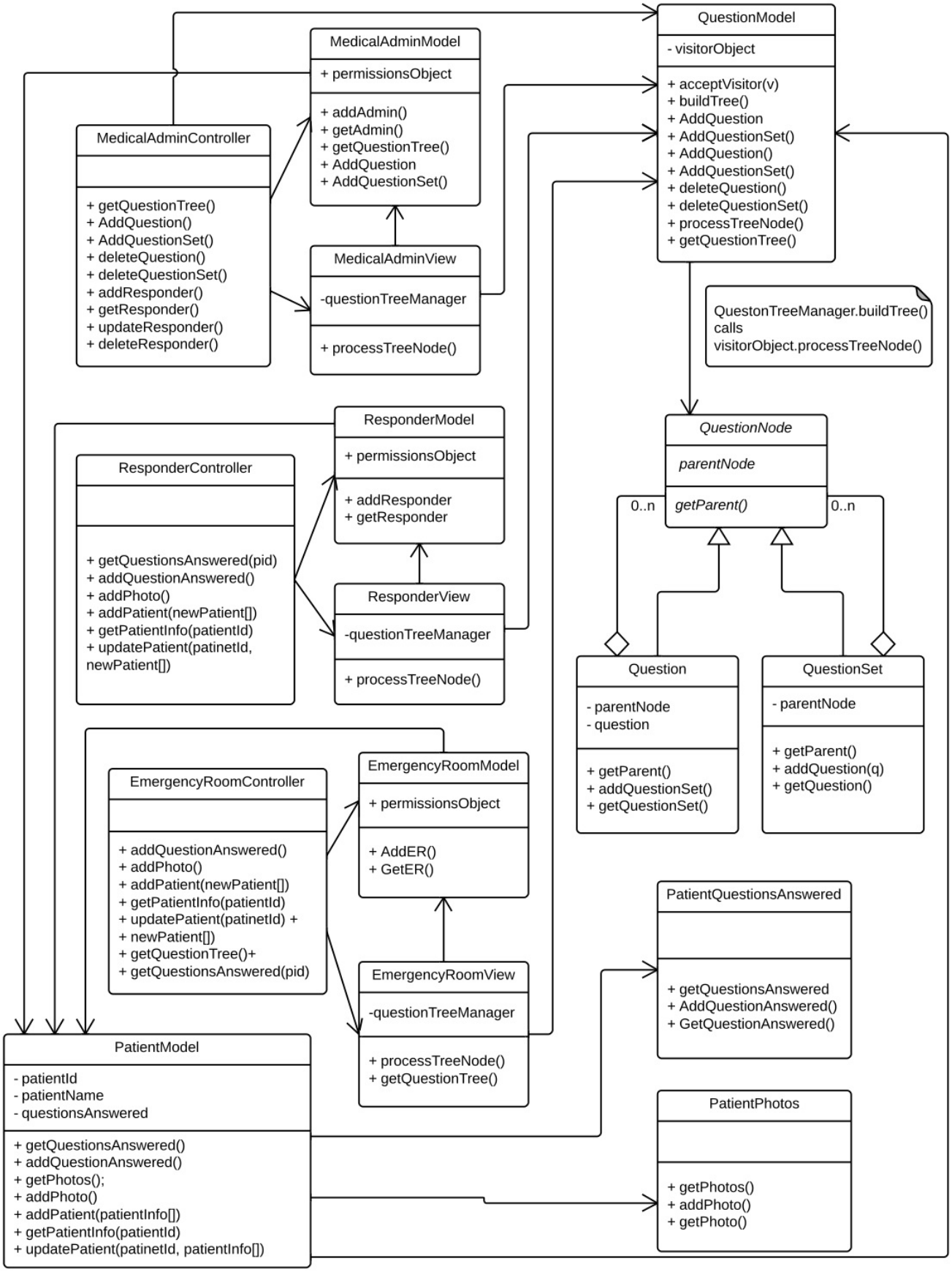
Interaction #2: The sequence diagram below outlines the process that occurs when the emergency responder responds with a boolean “True” to a question from the question page. When this happens, the framework will receive the HTTP request from the user’s browser. The framework will then communicate with the ResponderController, which passes on the request to the ResponderModel. The ResponderModel notifies the ResponderController that it was successful, and this is passed on to the Framework, and the user’s browser respectively.

Emergency Responder Marks a Question True for a Patient



Interaction #3: The sequence diagram below describes the process that occurs when the emergency room's browser polls to check for updated information. When this happens, the framework will receive the HTTP request from the user's browser. The framework will communicate with the EmergencyRoomController, which passes on the request to the EmergencyRoomModel. The EmergencyRoomModel returns the response to the EmergencyRoomController, the Framework, and the user's browser in that respective order.

Emergency Room User's browser polls to see if needs to refresh patent's answered questions



Class Diagram