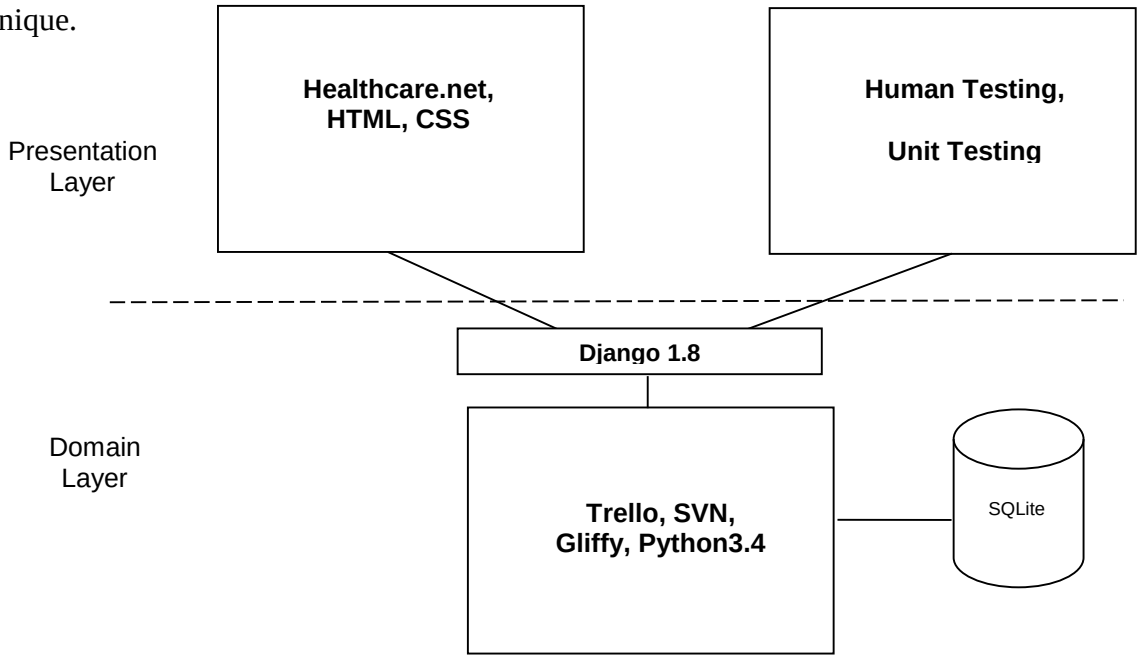
Product Design

Team StackOverflowGooglers

Revision Number	*Revision Date*	*Summary of Changes*	*Author(s)*
1.0	*09/21/2015*	*Filling out the whole thing*	*Doug Gawne, Paul Hulbert, Lucas Smith*
2.0	*10/22/15*	*Updated Design Rationale and Renamed Use Cases*	*Lucas Smith*
2.1	*10/26/15*	*Updated Components/Functions, Use Cases, and Sequence Diagrams for R2*	*Lucas Smith*

Architectural Model

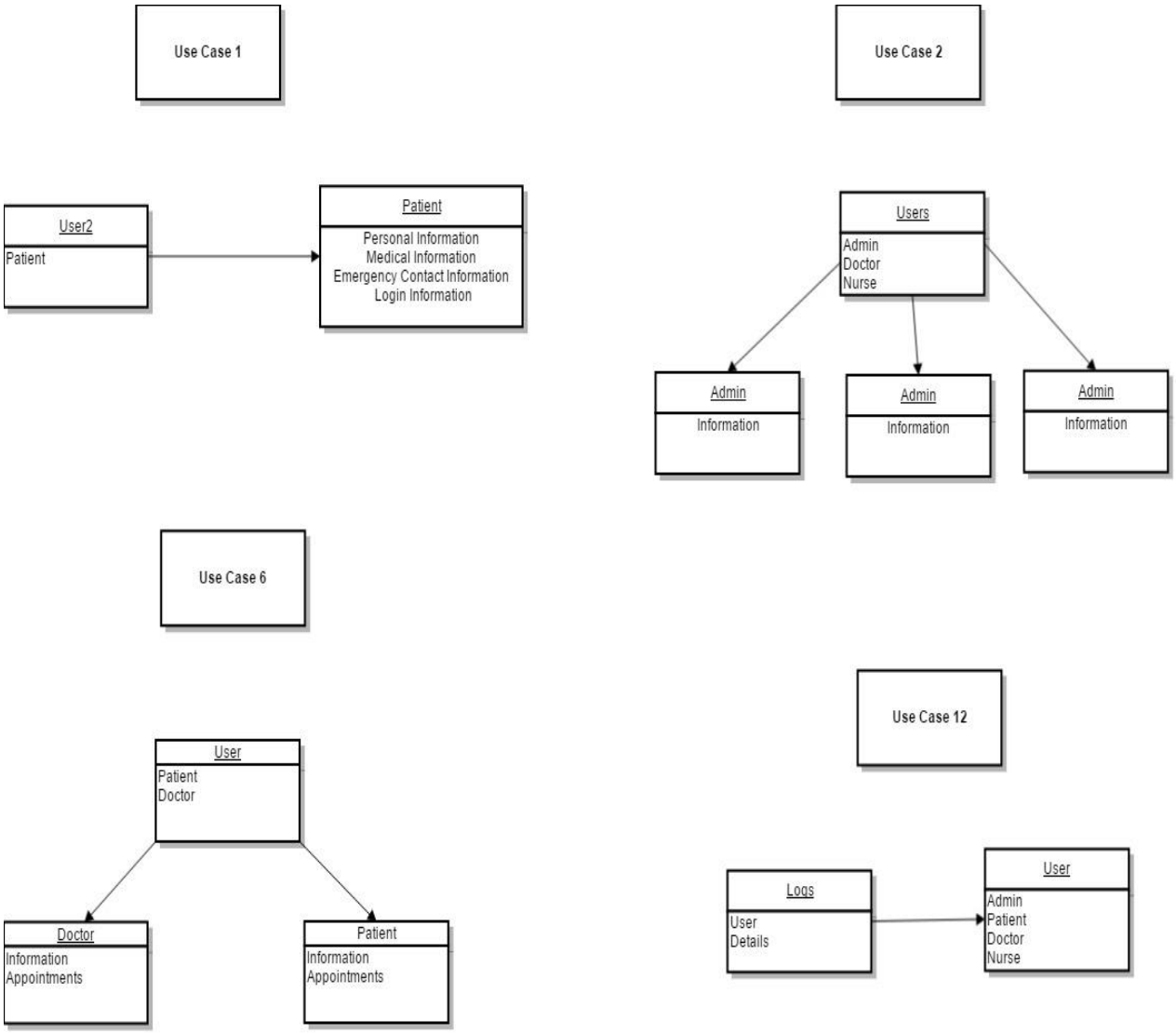
This diagram represents the major subsystems of the product. Initially focus on the domain layer and its components before decomposing the user interface component. Note that a common interface allows both the GUI and a Command Line Interface to access the domain model in the same manner without regard to the type of presentation technique.

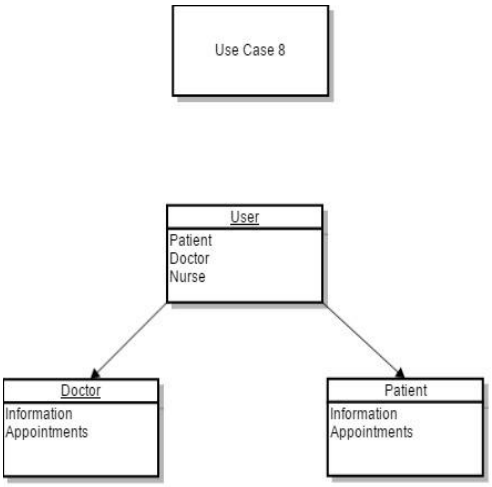


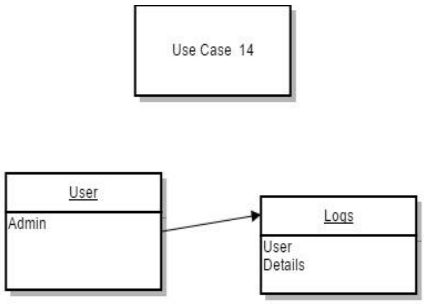
Components and Functions

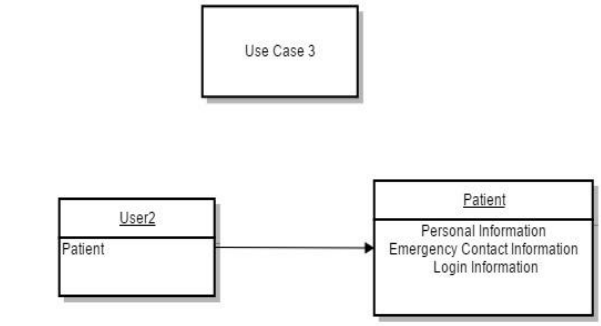
Home Page	• Patient login link • Doctor login link • Nurse login link • Admin login link • Admin News Feed
Patient Login	• Patient login form • Patient register link
Admin Login	• Superuser functionality
Doctor Login	• Doctor login form
Nurse Login	• Nurse login form
Doctor/patient create appointment	• Cancel link • details form
Doctor/patient view calendar	• logout link • create appointment link • edit patient details link
Nurse view calendar	• logout link • shows all appointments over a week • edit appointment link
Patient edit patient details	• cancel link • details form
Doctor/patient/nurse edit appointment	• cancel button for doctor/patient • detail form
Admin view logs	• Superuser can view models with logs contained
Admin View Stats	• Admin viewing the statistics of the system
Doctor/Nurse Edit Patient Medical info	• View Patient Medical Info • View Prescription
Doctor functions for a patient	• Add/Remove Prescription • Evaluate/Release Test Results • Upload files into patient Medical Info
Patient View Test Results	• View results of a test on medical info page • View discharge or extension of stay • Transfer of hospitals
Messaging	• Doctor/Nurse/Admin/Patient private messaging • Admin adding message of the day

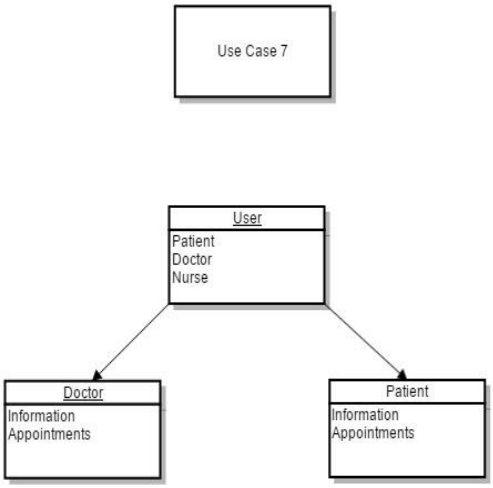
Class Diagram(s)

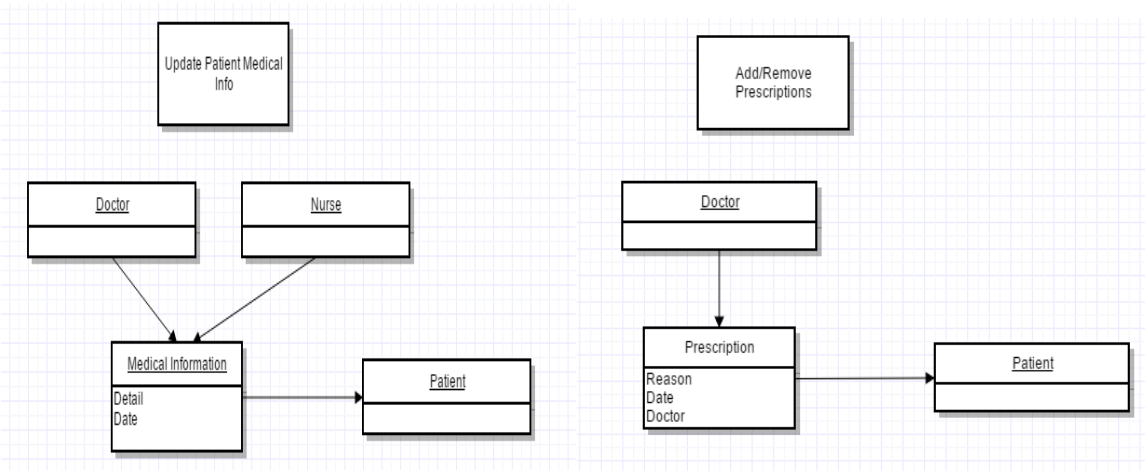


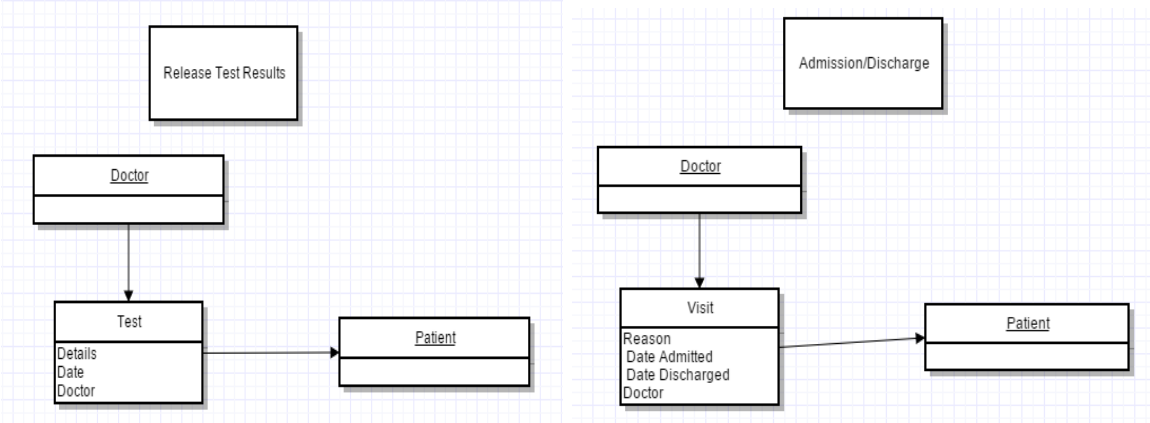

Use Case 8
User
Patient
Doctor
Nurse
Doctor
Information
Appointments
Patient
Information
Appointments


Use Case 14
User
Admin
Logs
User
Details


Use Case 3
User2
Patient
Patient
Personal Information
Emergency Contact Information
Login Information

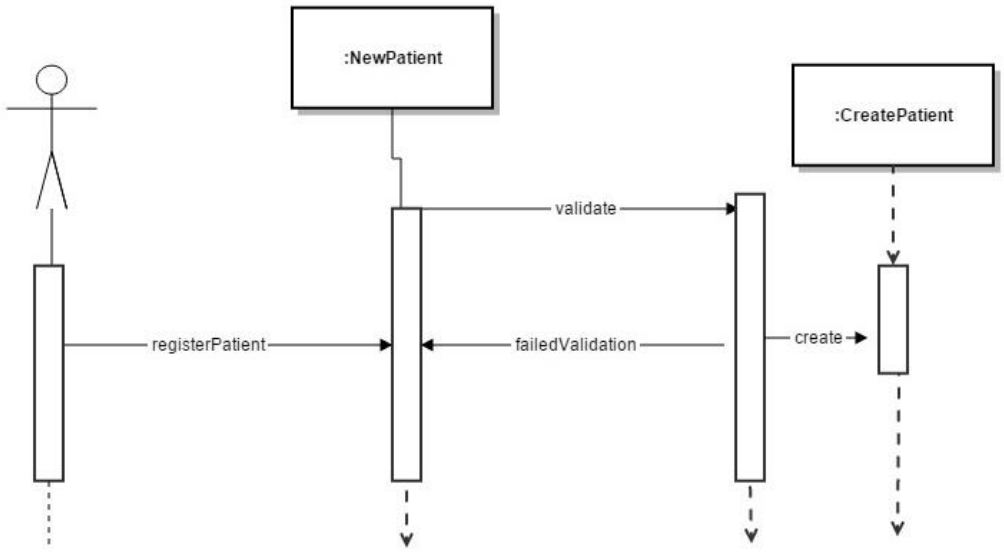

Use Case 7
User
Patient
Doctor
Nurse
Doctor
Information
Appointments
Patient
Information
Appointments


Update Patient Medical Info
Doctor
Nurse
Medical Information
Detail
Date
Patient
Add/Remove Prescriptions
Doctor
Prescription
Reason
Date
Doctor
Patient

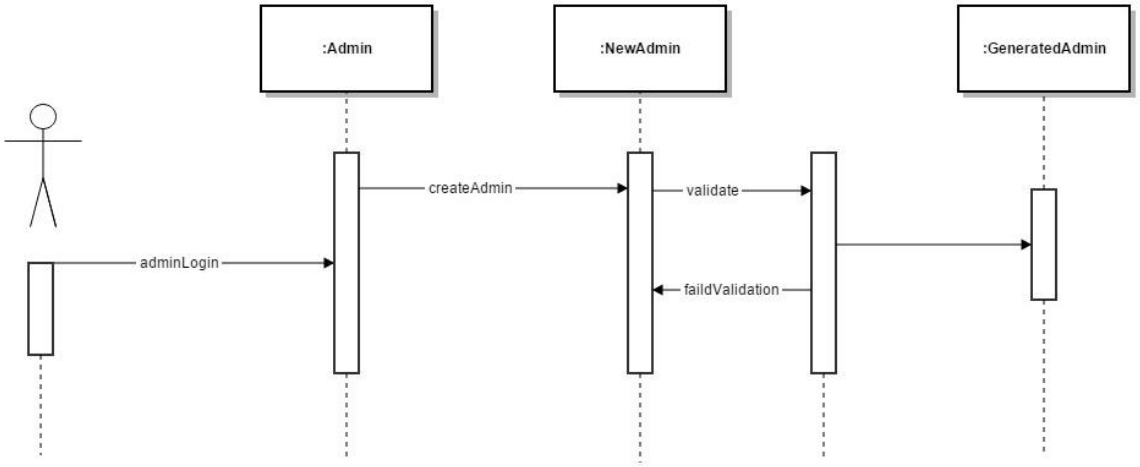


Sequence Diagram(s)

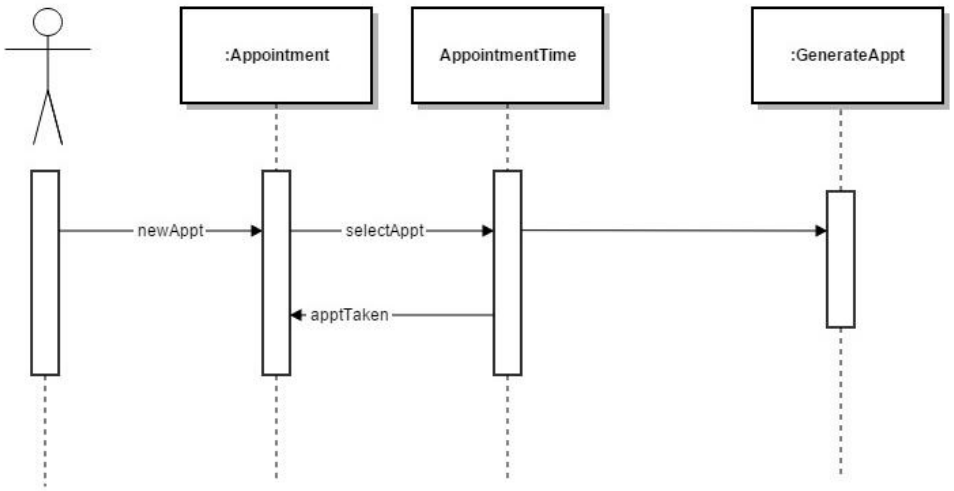
Registration-Use Case 1



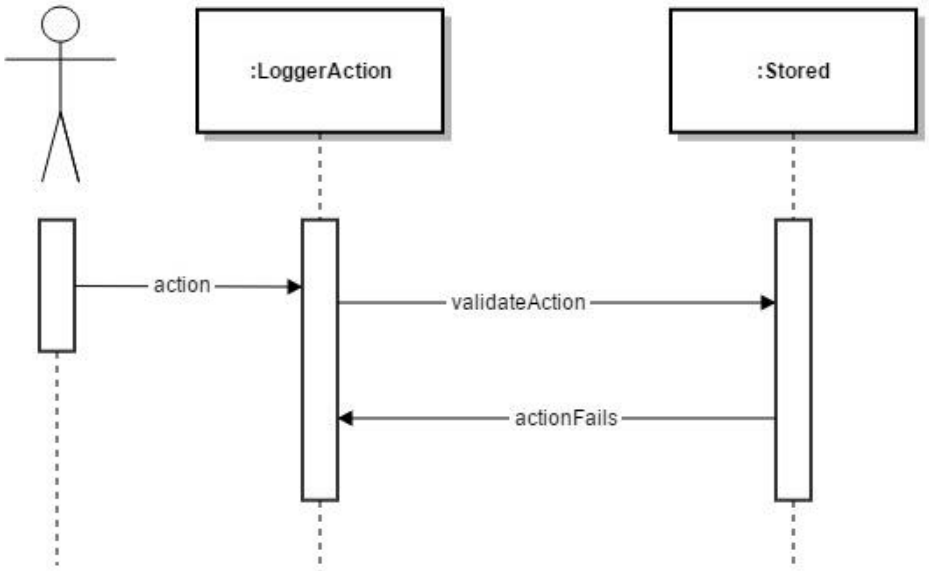
Admin Registration- Use Case 2



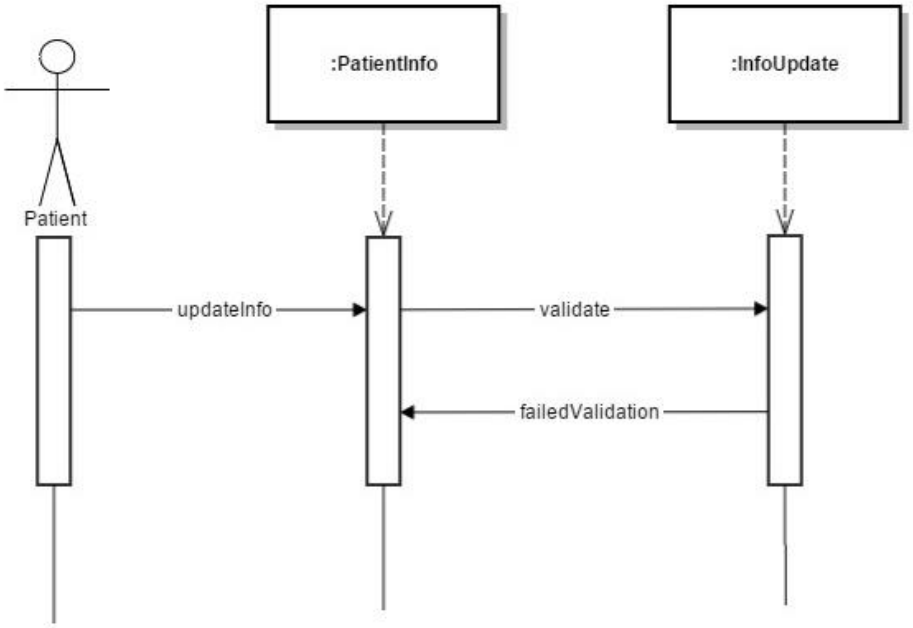
Create Appointment- Use Case 6



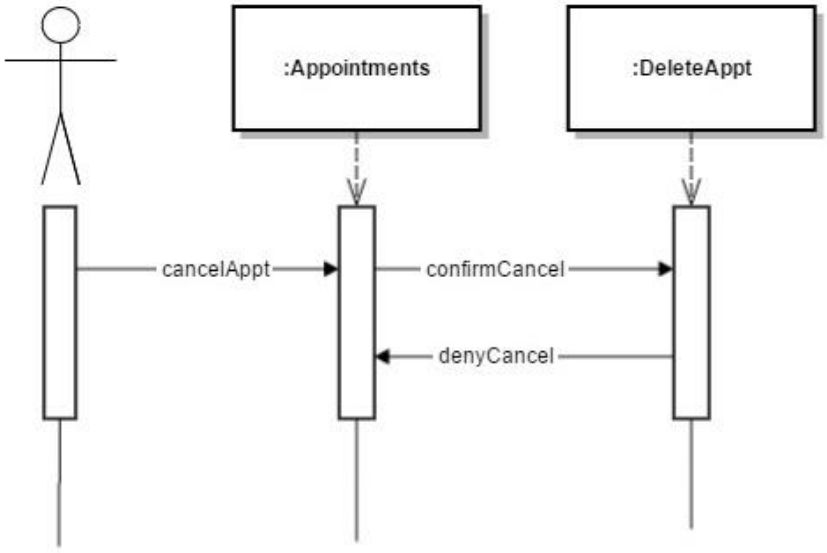
Logging Activity- Use Case 12



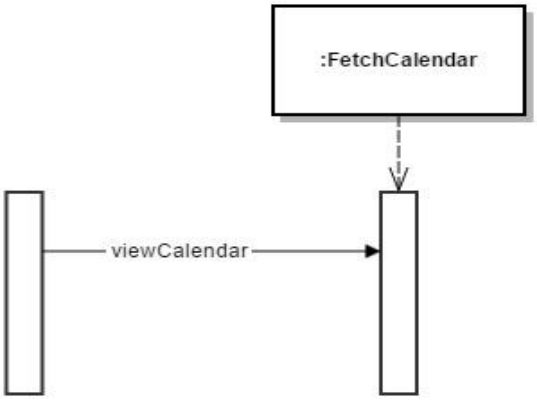
Update Patient Profile Information- Use Case 6



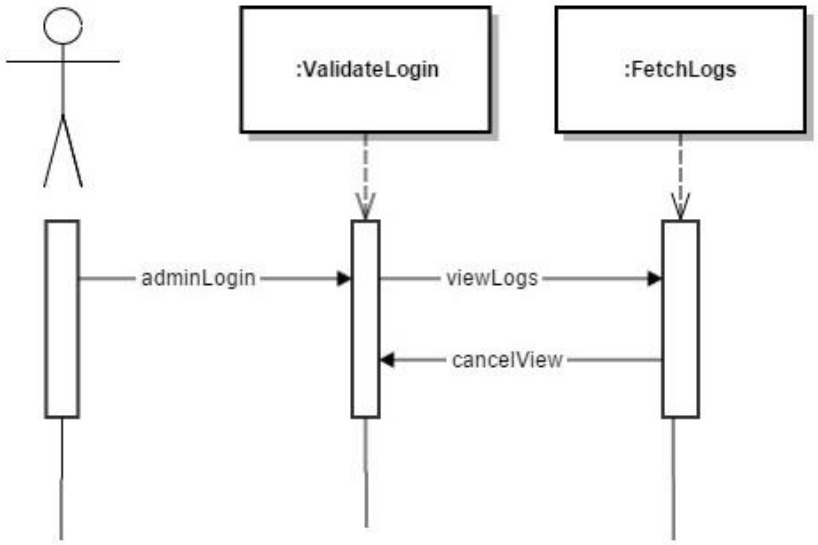
Cancel Patient Appointment- Use Case 14



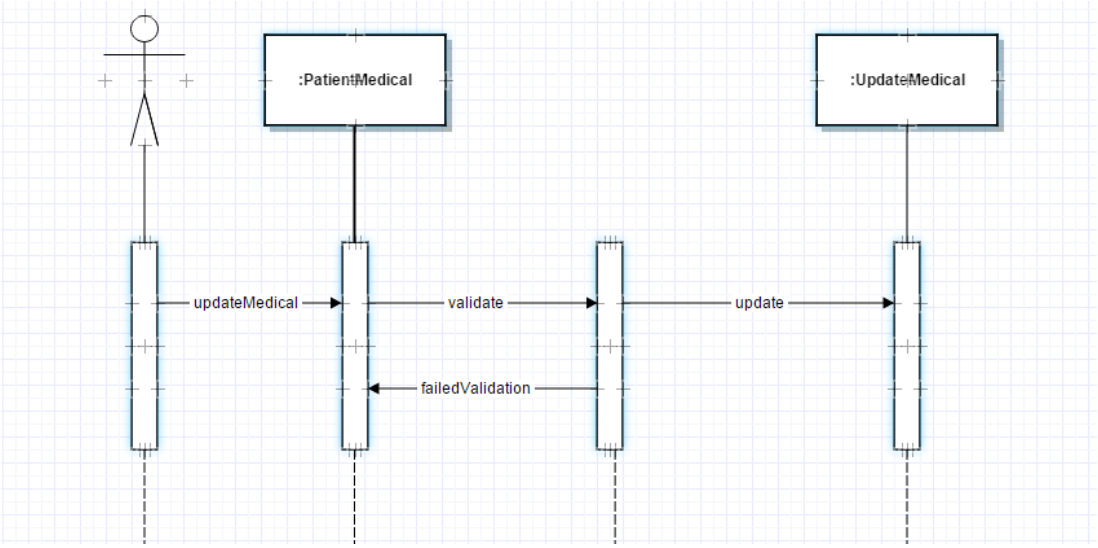
Appointment Calendar- Use Case 3



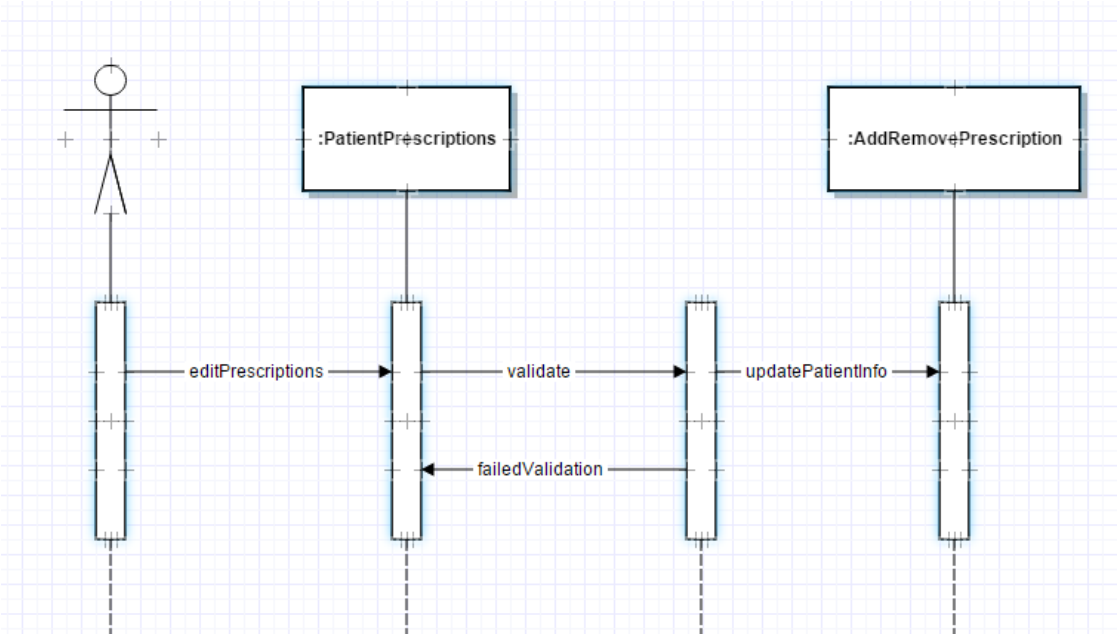
Viewing Activity Log- Use Case 7



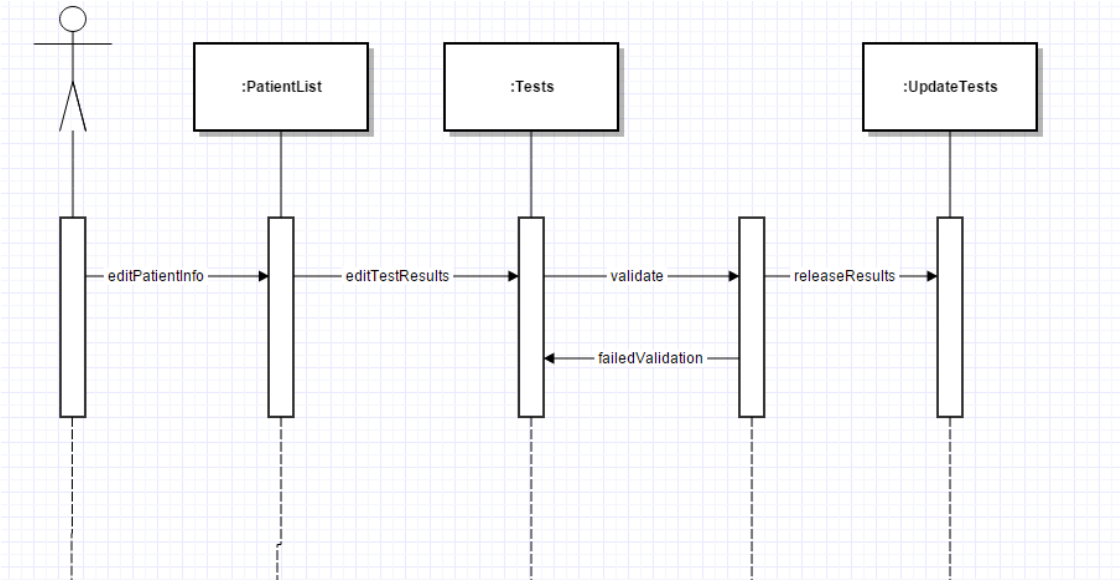
Update Patient Medical Info



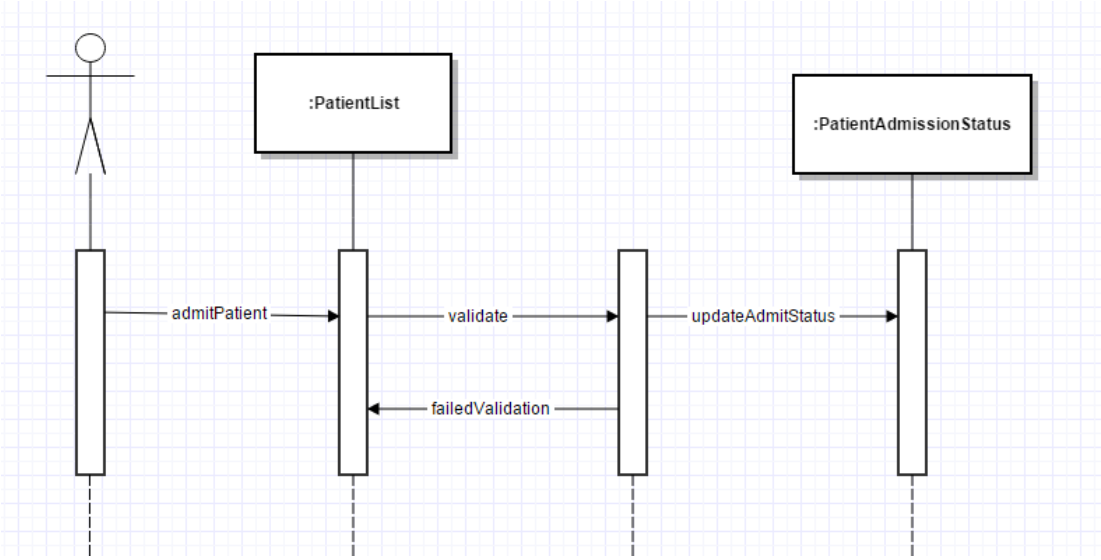
Add/Remove Prescription



Release Test Results



Admission/Discharge



Design Rationale

We chose to use only one app for this project. We could not find any benefit aside from namespace collisions that would result from making multiple apps.

This meant we had to figure out a different way to prevent namespace collisions. What we chose to do was create an individual .py file for each view, and import them into the main views.py file. This also let us be able to work on views simultaneously without risk of merge conflicts.

We chose to use the Django superuser as the admin, because it gives us everything we need, custom built. Perhaps we will edit the templates later to make it work better, but for now it works well.

We also broke the work into front end, back end (views), and models. This lets us break into smaller groups to do the different parts.

Django has a built-in login view, but we chose to make our own because it allows us to better control it and prevent someone from re-logging in.

We chose to create our own calendar because it gave us more control over how it appeared to the user.

We also replaced the Django superuser with custom built views to make it more useable.