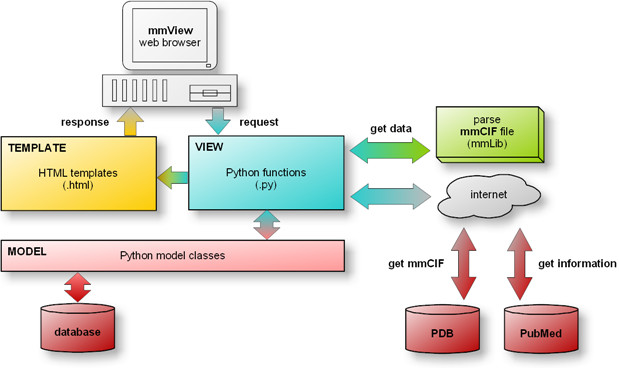
**Product Design**

|  |  |
| --- | --- |
| **Team** | **The Procrastinators** |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Revision Number*** | ***Revision Date*** | ***Summary of Changes*** | ***Author(s)*** |
| *0.1* | *02/22/16* | *Initial Changes* | *Chris Lim* |
| ***0.2*** | ***02/29/16*** | ***Settled on basic class design, fixed some UML diagrams*** | ***Maximillian McMullen*** |
| ***0.3*** | ***03/02/16*** | ***Minor changes to core class structure, decided on external tools/apis*** | ***Maximillian McMullen*** |
| ***0.4*** | ***03/04/16*** | ***Major class changes and project structure, but primary functionality is unchanged*** | ***Maximillian McMullen*** |
| ***0.5*** | ***03/06/16*** | ***Minor changes to class structure, last minute changes to apis used, final design for views and general aesthetics*** | ***Maximillian McMullen*** |
| ***0.6*** | ***03/07/16*** | ***Fixed remaining diagrams*** | ***Maximillian McMullen*** |
| ***0.7*** | ***04/02/16*** | ***Modified components to reflect R2*** | ***Maximillian McMullen*** |
| ***0.8*** | ***05/07/16*** | ***Added additional sequence diagrams*** | ***Maximillian McMullen*** |
| ***0.9*** | ***05/08/16*** | ***Created more sequence diagrams*** | ***Maximillian McMullen*** |
| ***1.0*** | ***05/08/16*** | ***Finalized R2 sequence diagrams*** | ***Maximillian McMullen*** |
| ***1.1*** | ***05/08/16*** | ***Revised R1 sequence diagrams*** | ***Maximillian McMullen*** |

# Architectural Model

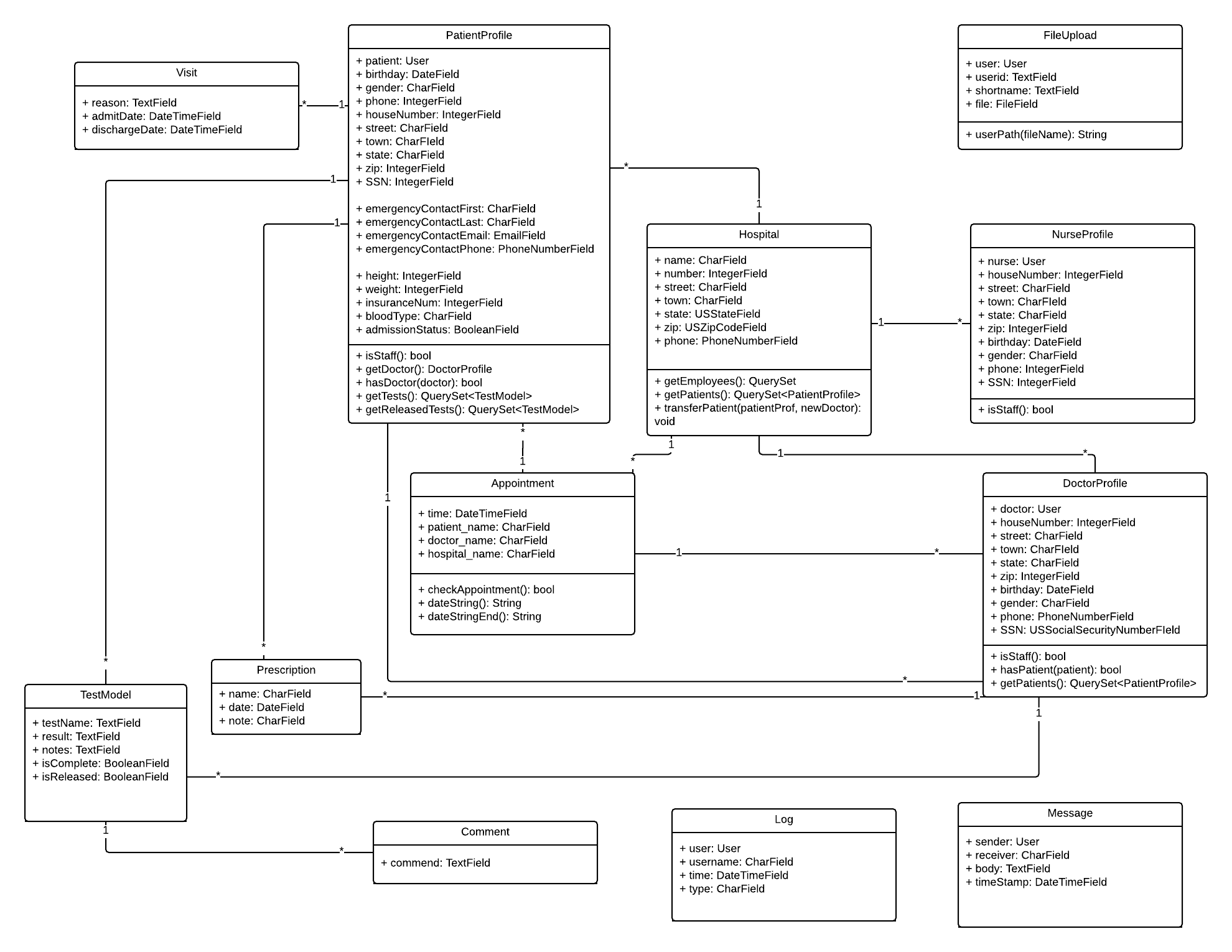
This diagram represents the major subsystems of the product. Initially focus on the domain layer and its components before decomposing the user interface component. Note that a common interface allows both the GUI and a Command Line Interface to access the domain model in the same manner without regard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



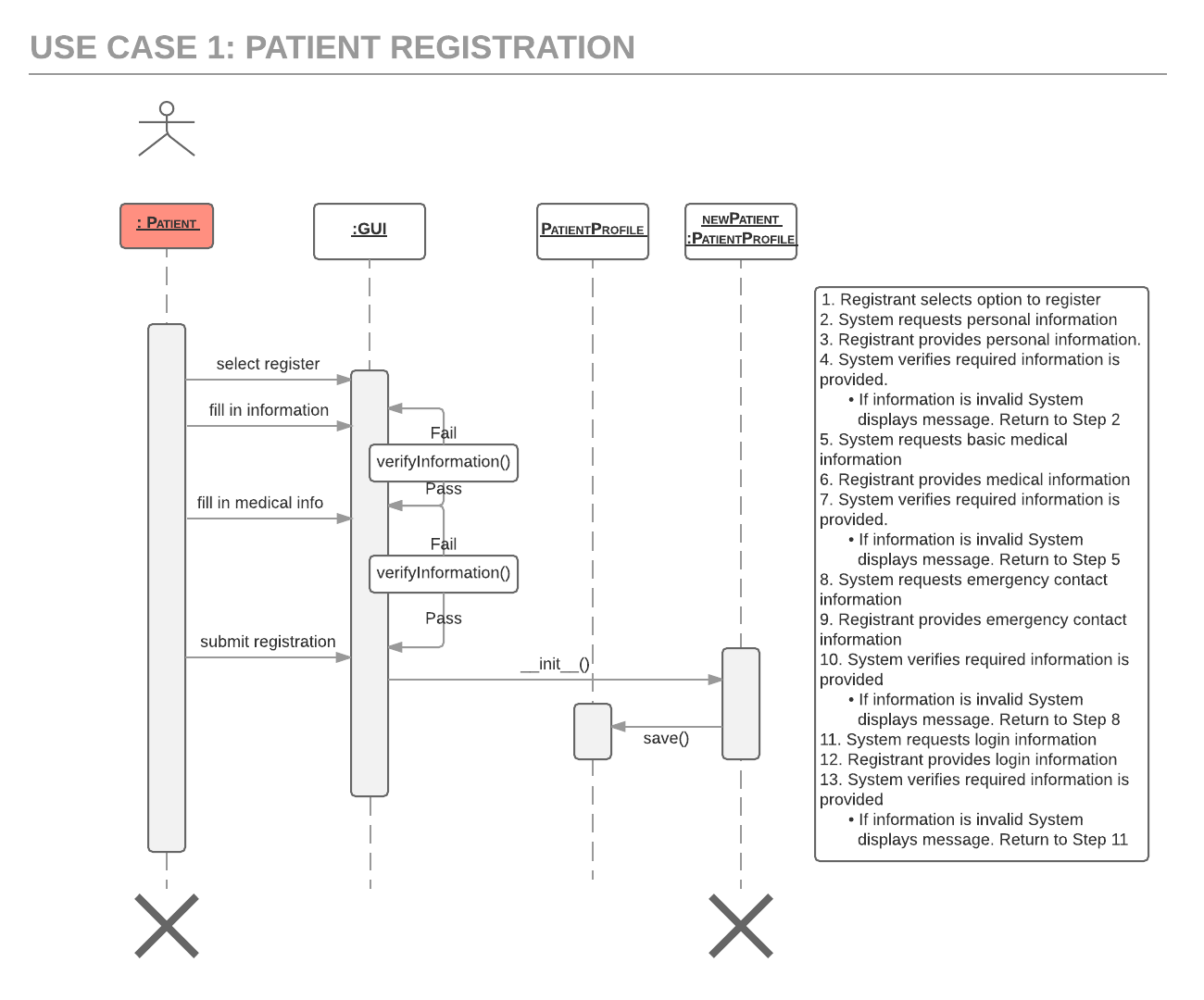
# Components and Functions

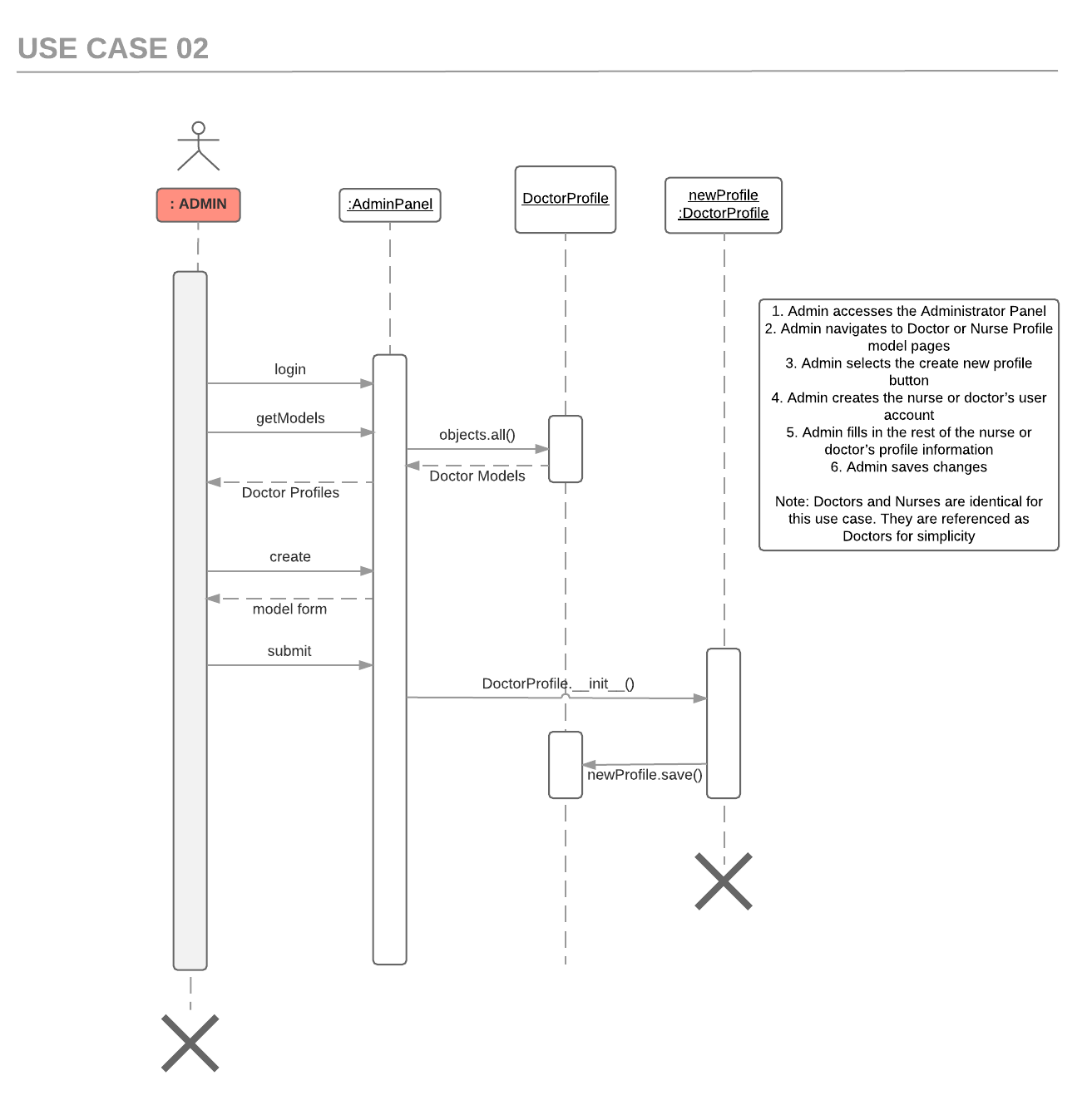
|  |  |
| --- | --- |
| Patient Registration | * Component State: maintains:   + Name   + Phone Number   + Email   + Insurance   + Emergency Contact   + Login Credentials * Component Behavior:   + Can be modified by “update patient account” |
| Admin Registration | * Component State: maintains:   + Name   + admin id * Component Behavior:   + Creates Doctors, Nurses, and other Admin accounts   + View Activity Log |
| View/Update Patient Account Information | * Component State: maintains   + personal information * Component Behavior:   + Updates patient account information   + Creates log events |
| View/Update Patient Medical Information | * Component State: maintains   + Medical information (test results, prescriptions, etc.) * Component Behavior:   + Updates medical information   + Creates log events |
| Logging System | * Component State: maintains   + Actions and activities run in the application |
| Appointments | * Component State: maintains   + Maintain time and date for appointments for Doctors and Patients * Component Behavior:   + Creates/Updates/Deletes appointment to be added to Calendar   + Checks to see if appointment is not conflicting with other appointments |
| Calendar | * Component State: maintains   + A visual list of the appointments * Component Behavior   + Can add appointments   + View appointments |
| Transfer Patients | * Component State: maintains   + Hospital transfers * Component Behavior   + Can transfer patients between doctors   + Can transfer patients between hospitals |
| Messaging | * Component State: maintains   + Text messages between users * Component Behaviour:   + Users can message other users   + Users can receive messages |

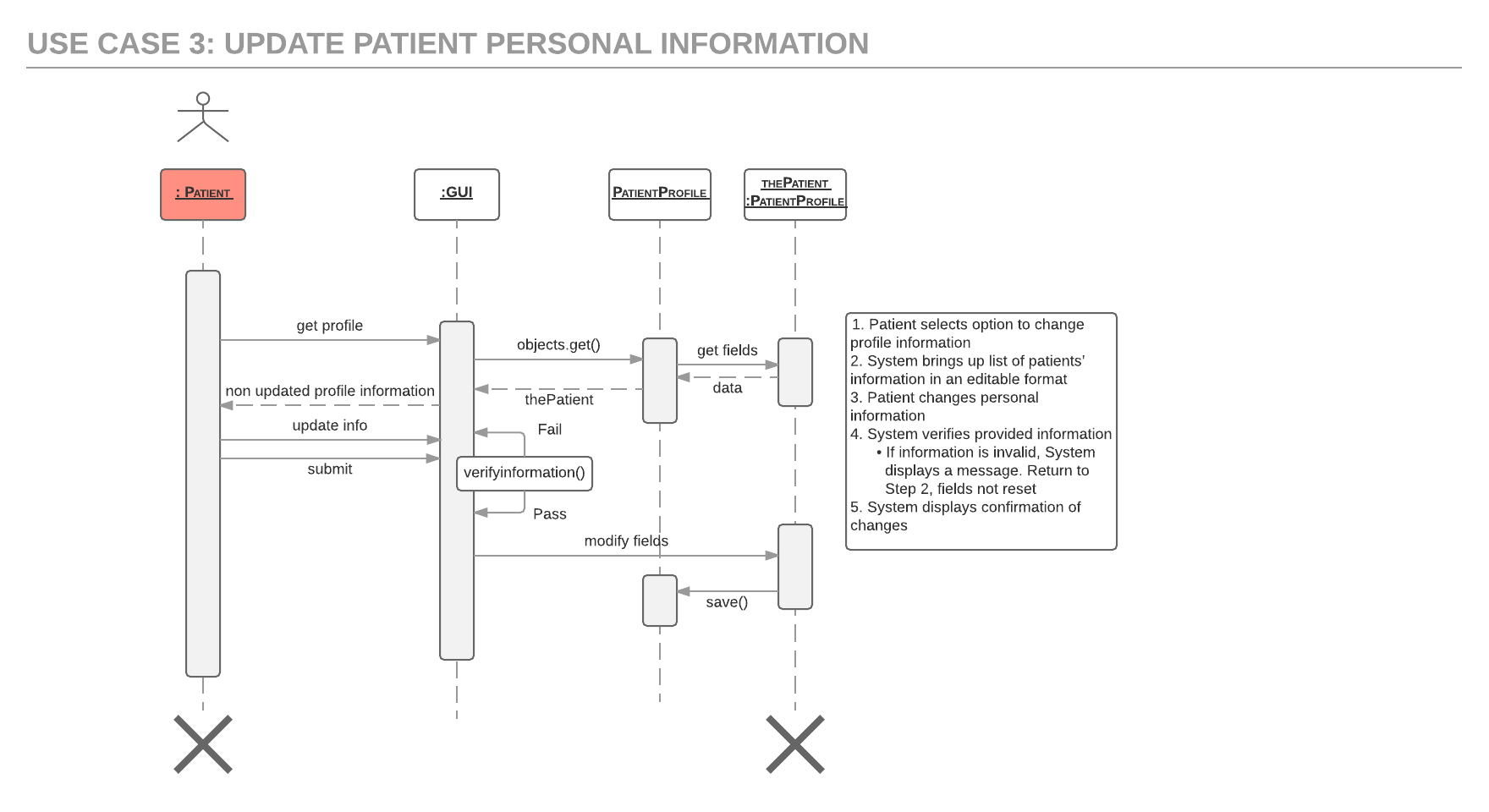
# Class Diagram(s)

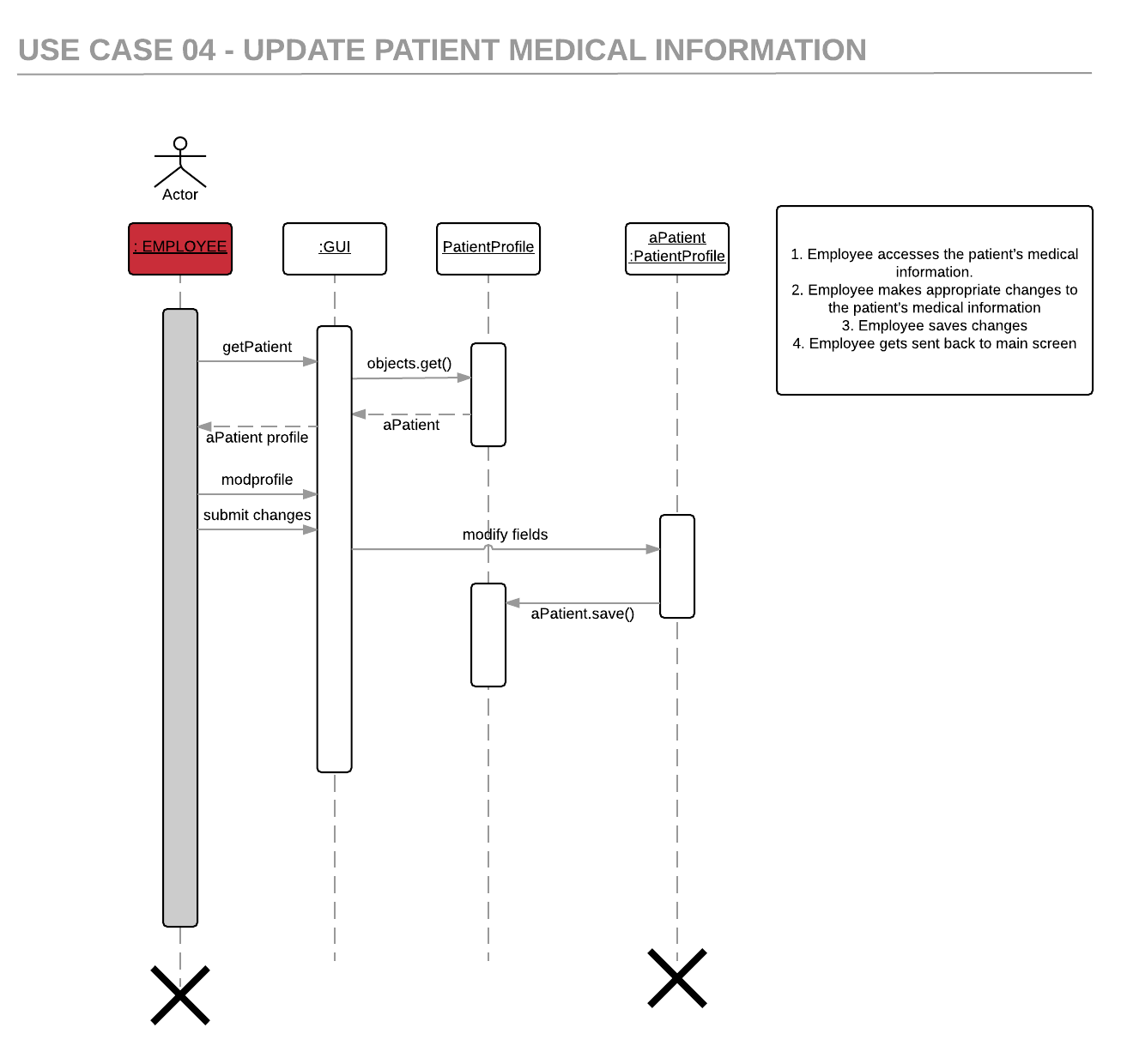


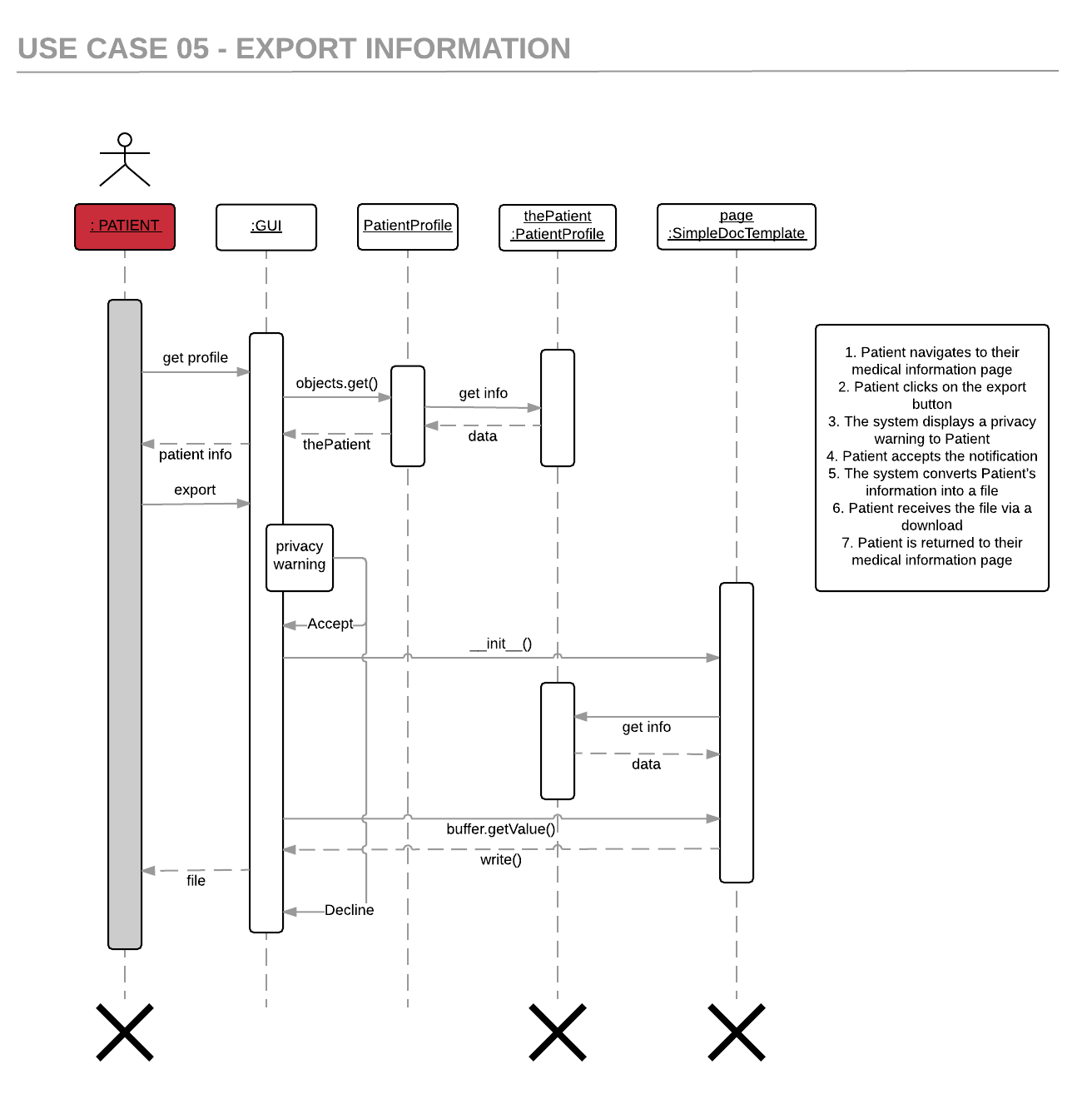
**Sequence Diagram(s)**

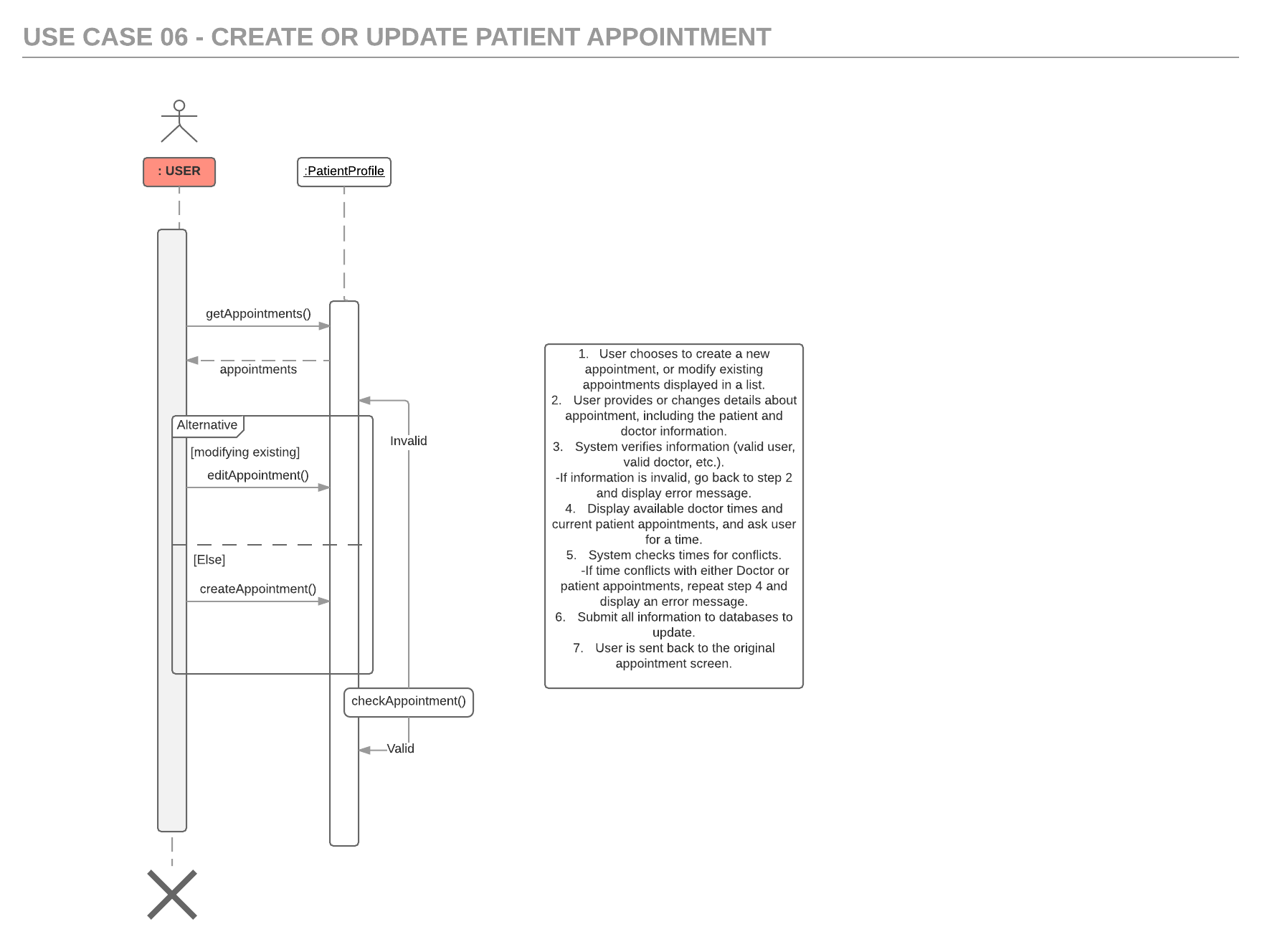


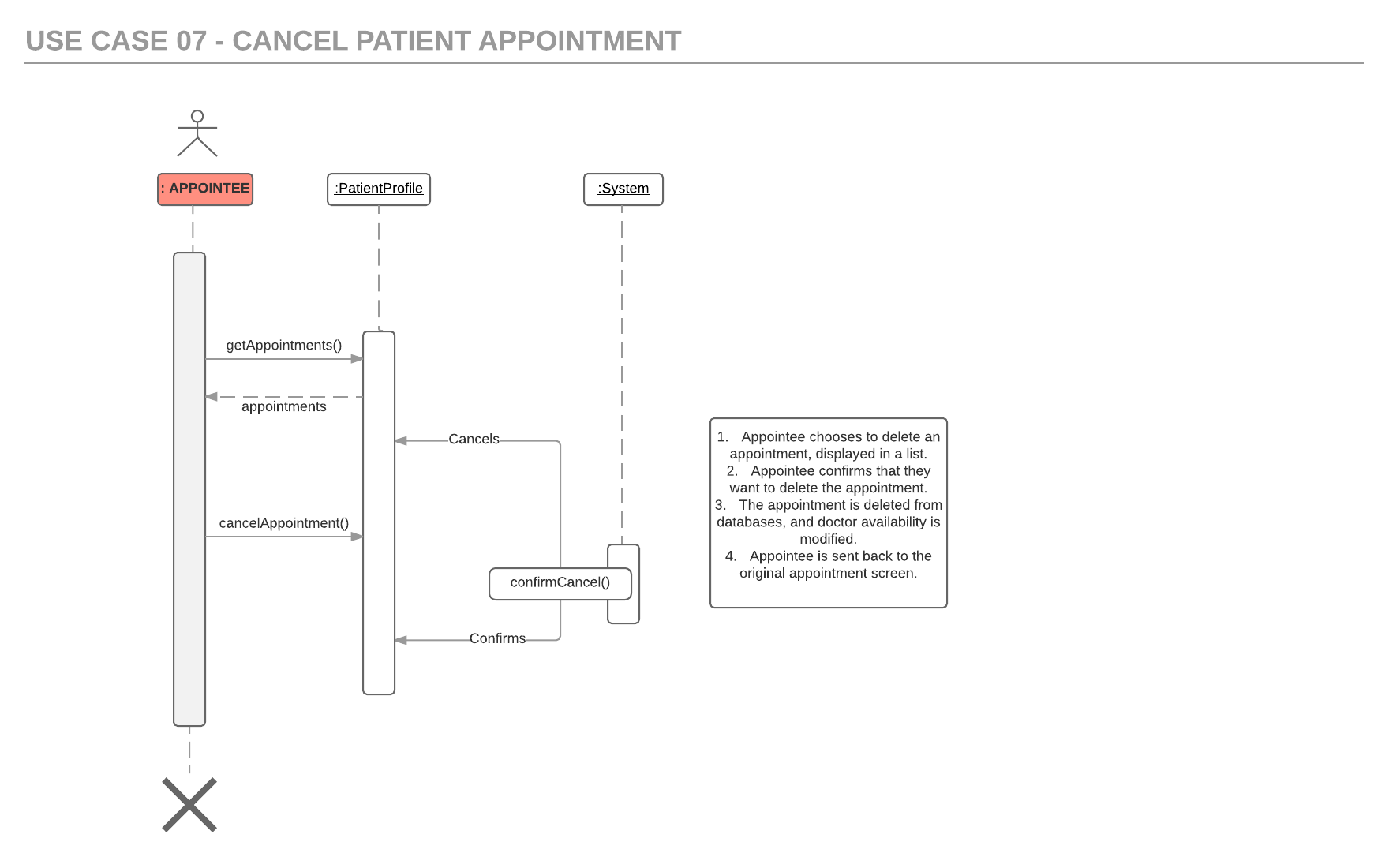


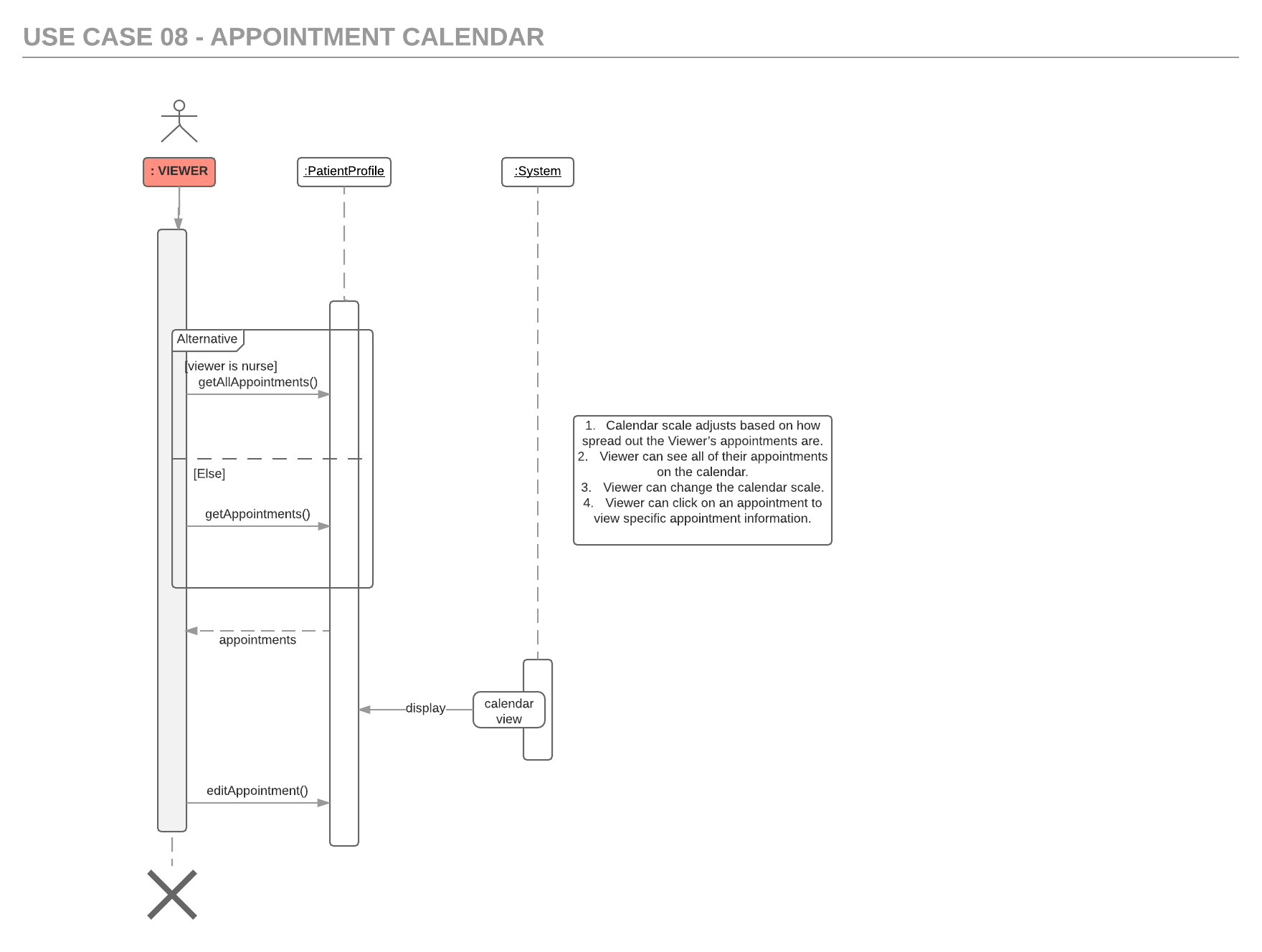


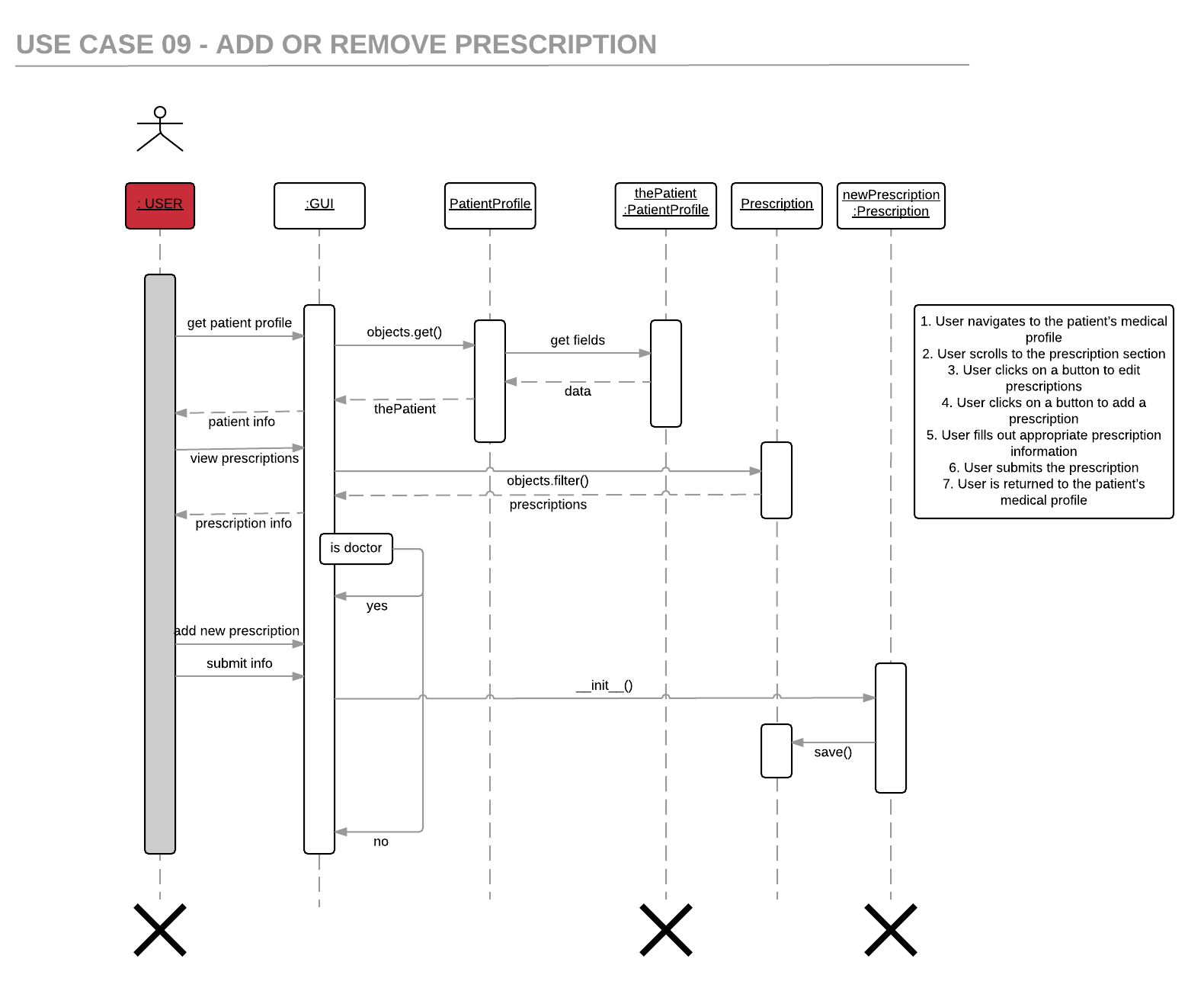


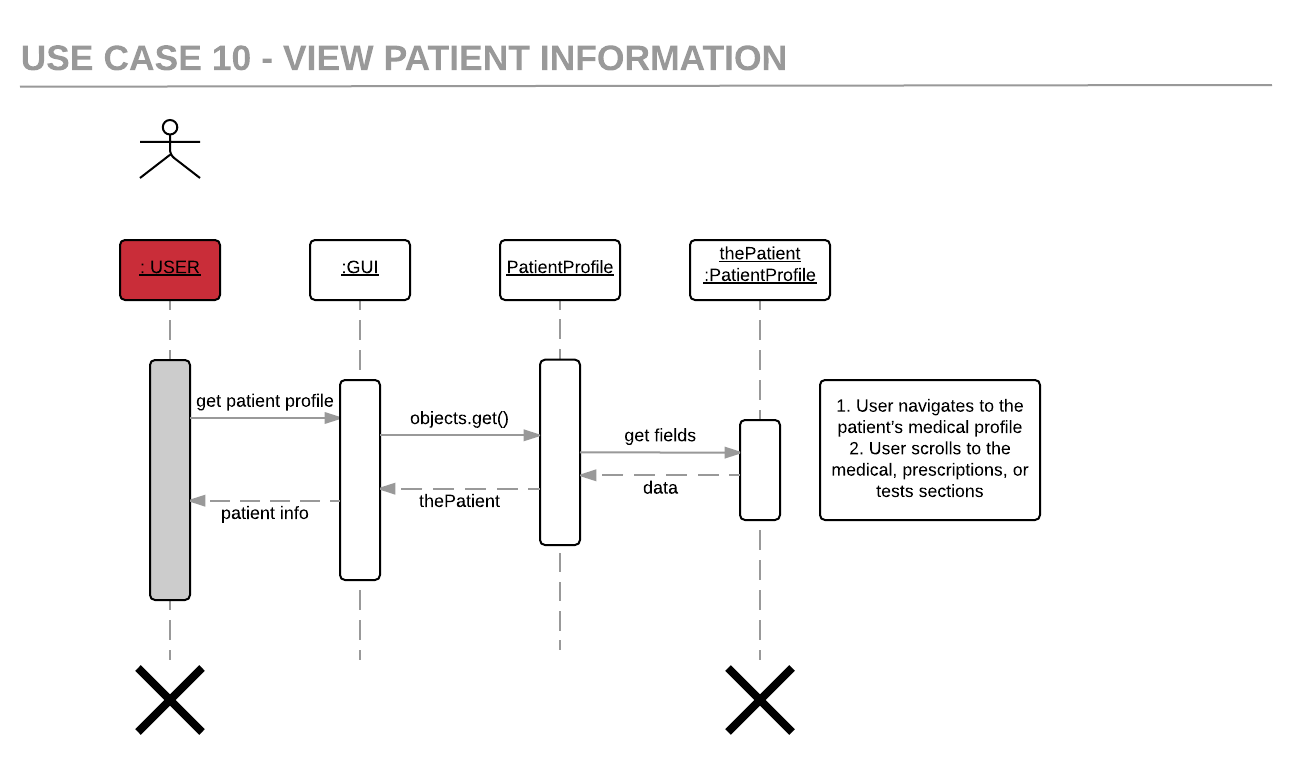


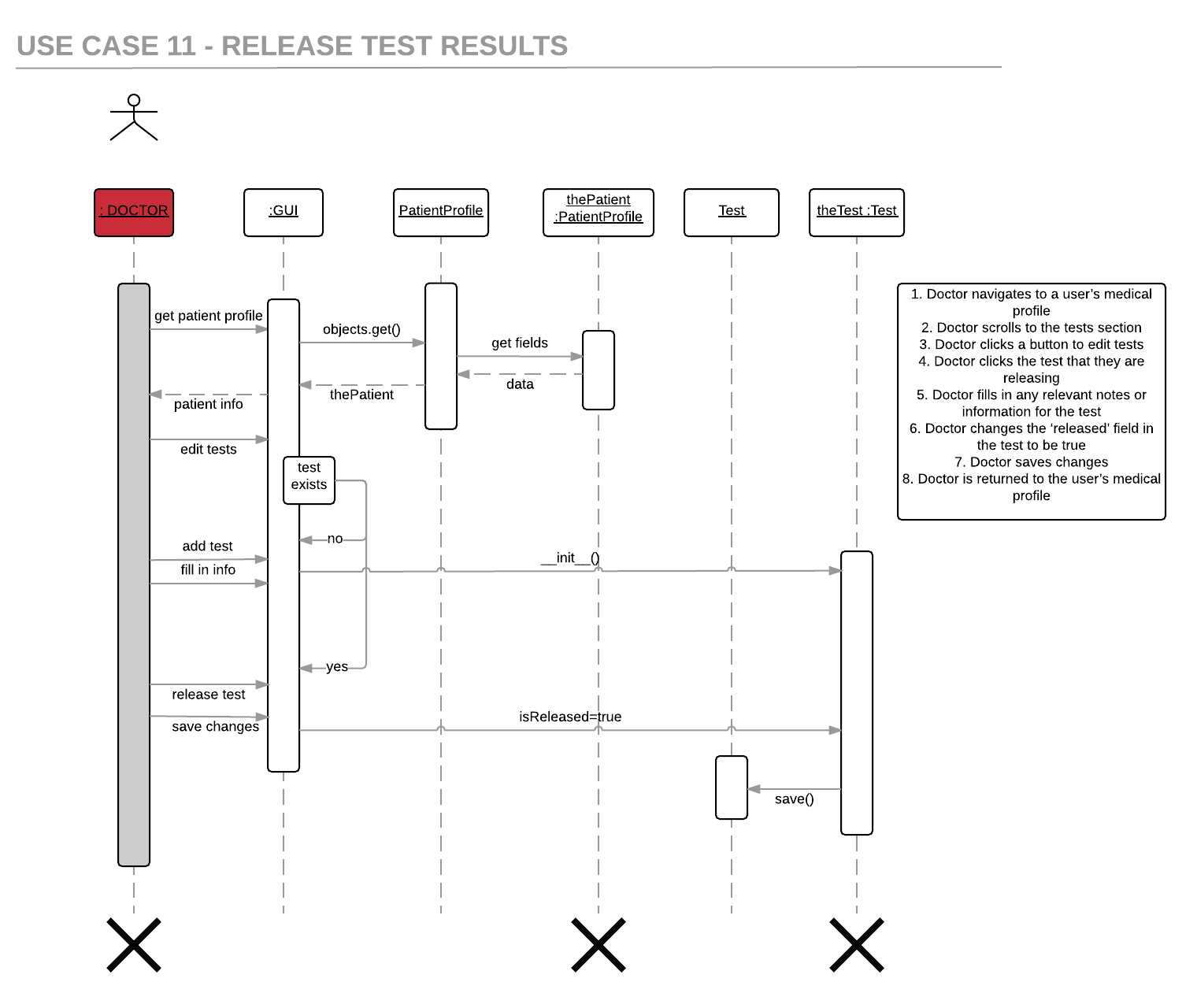


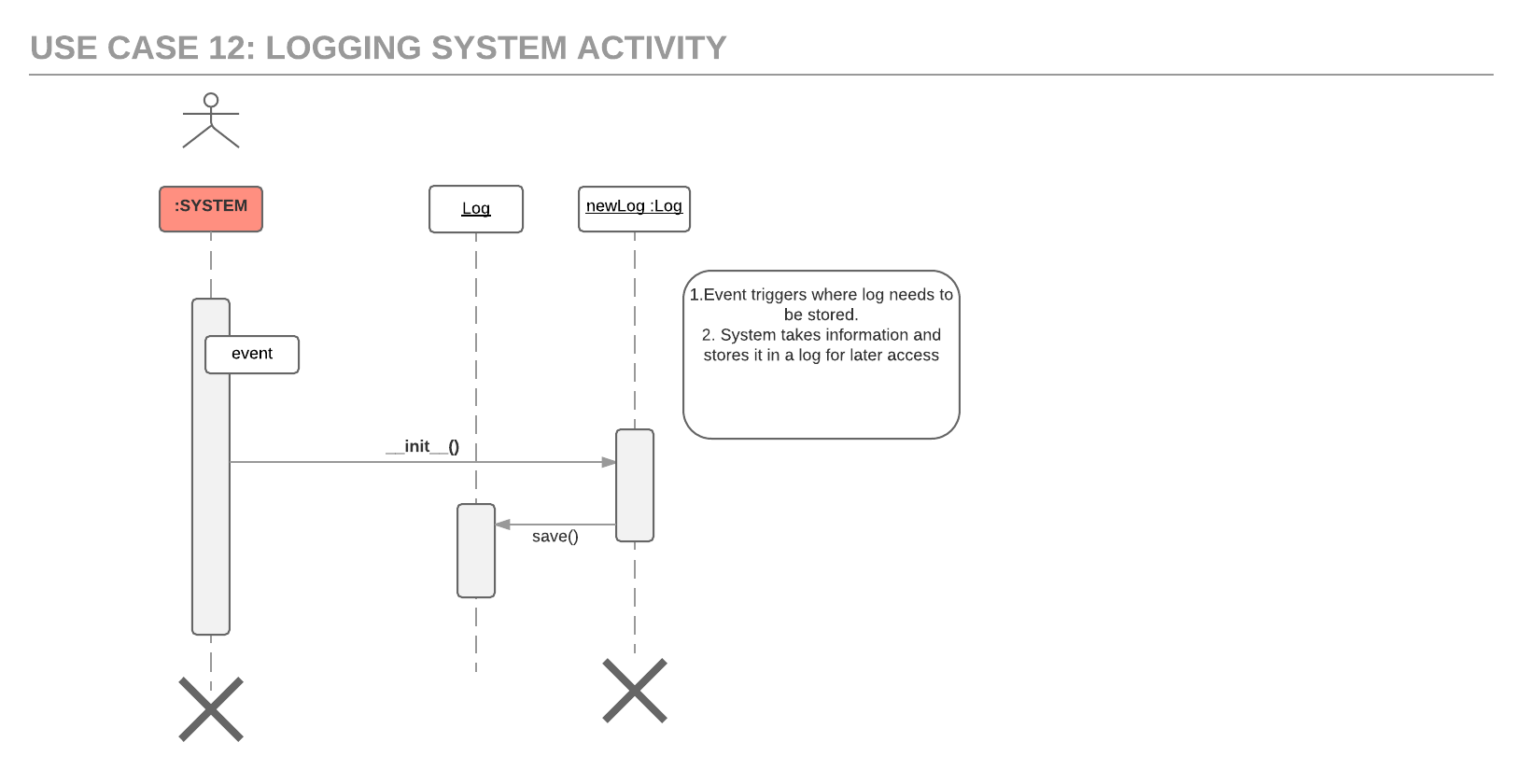


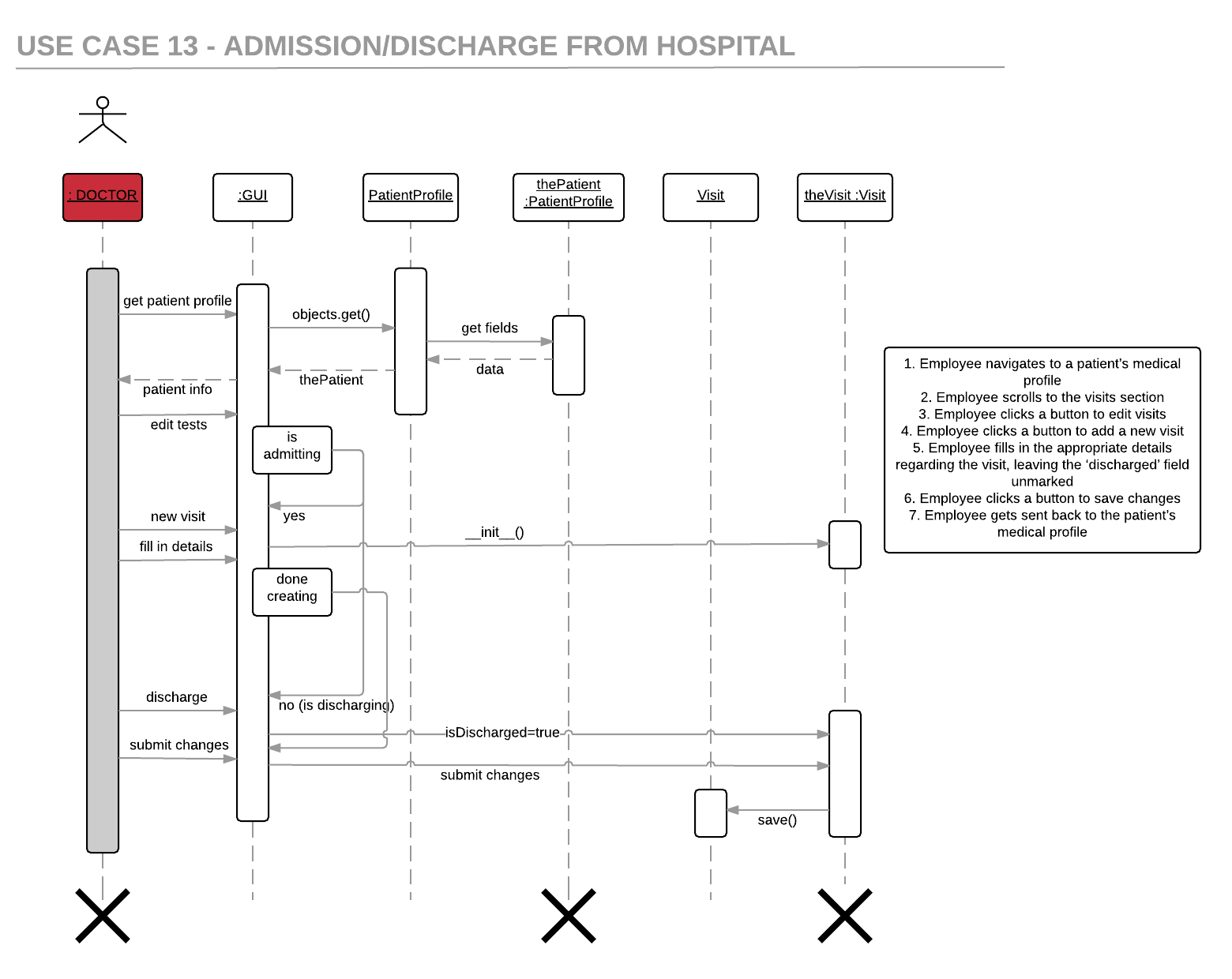


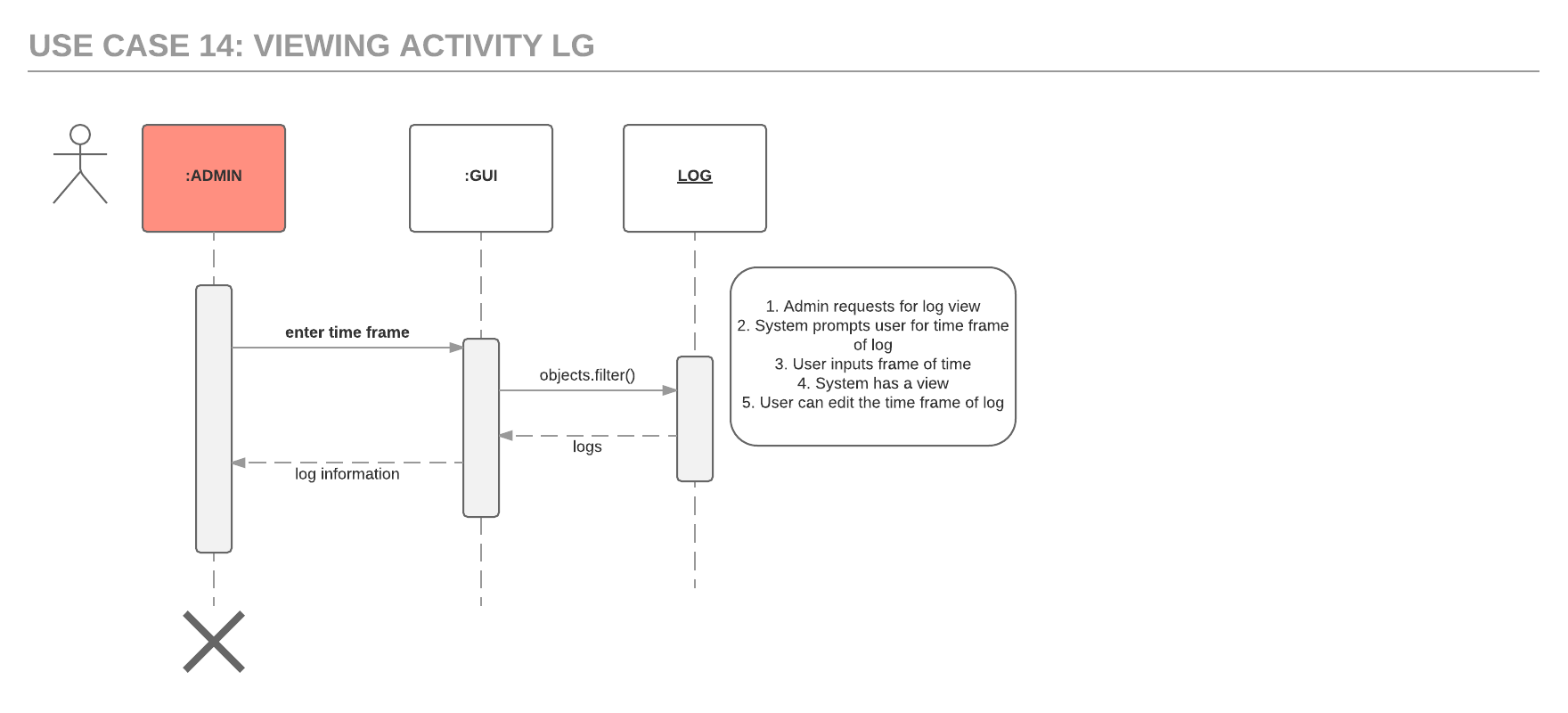


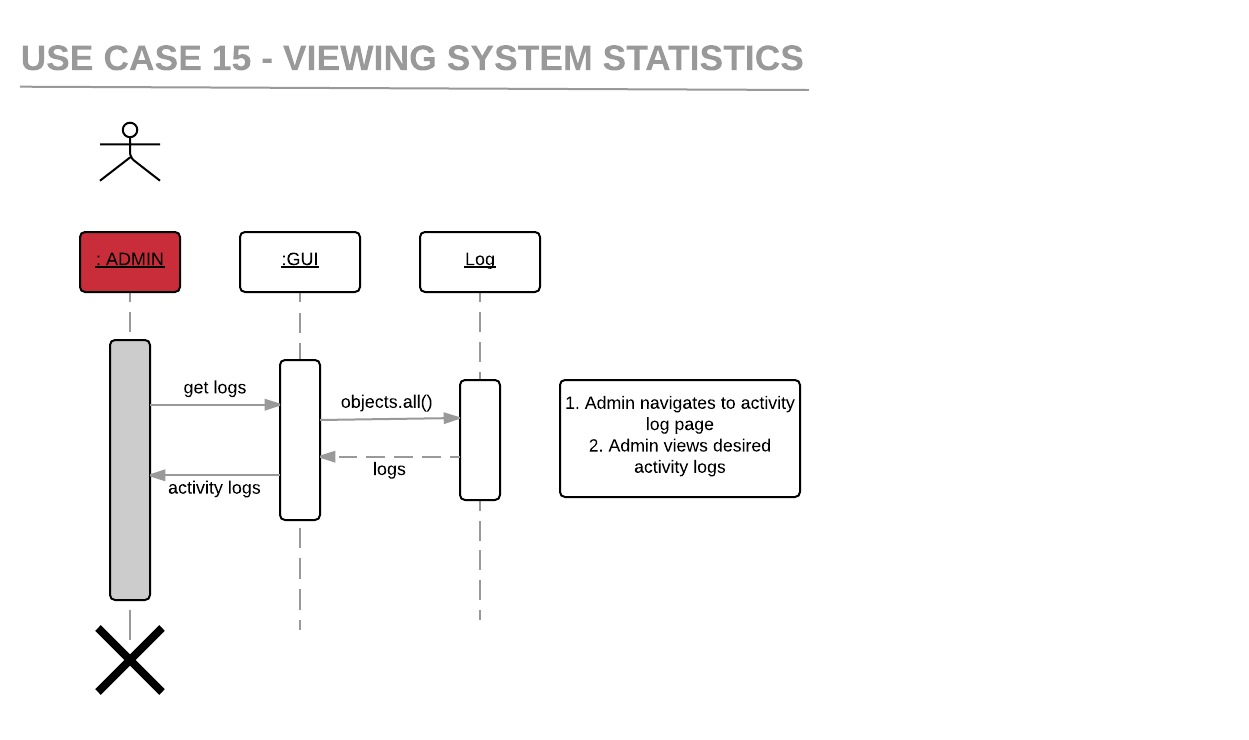


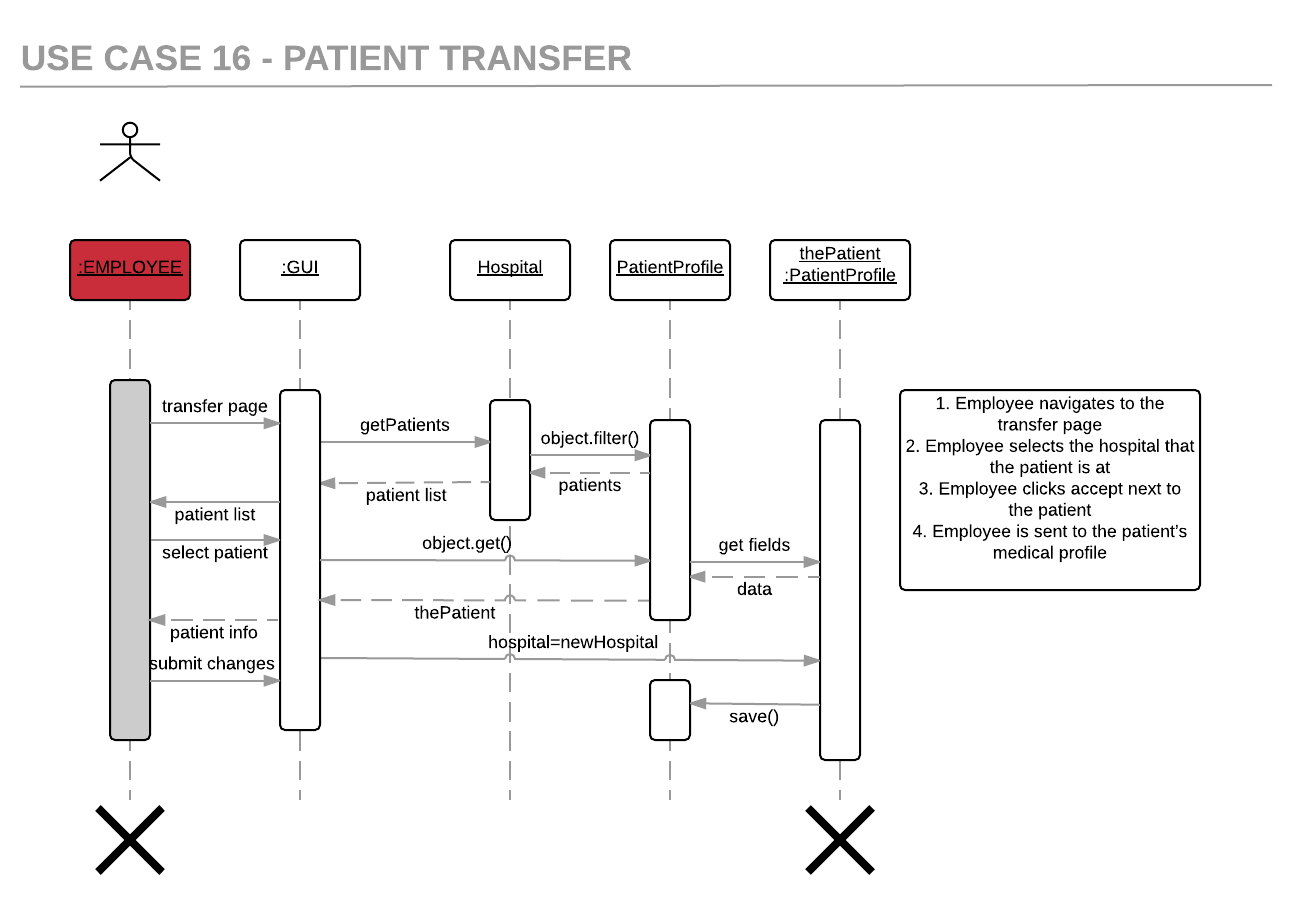


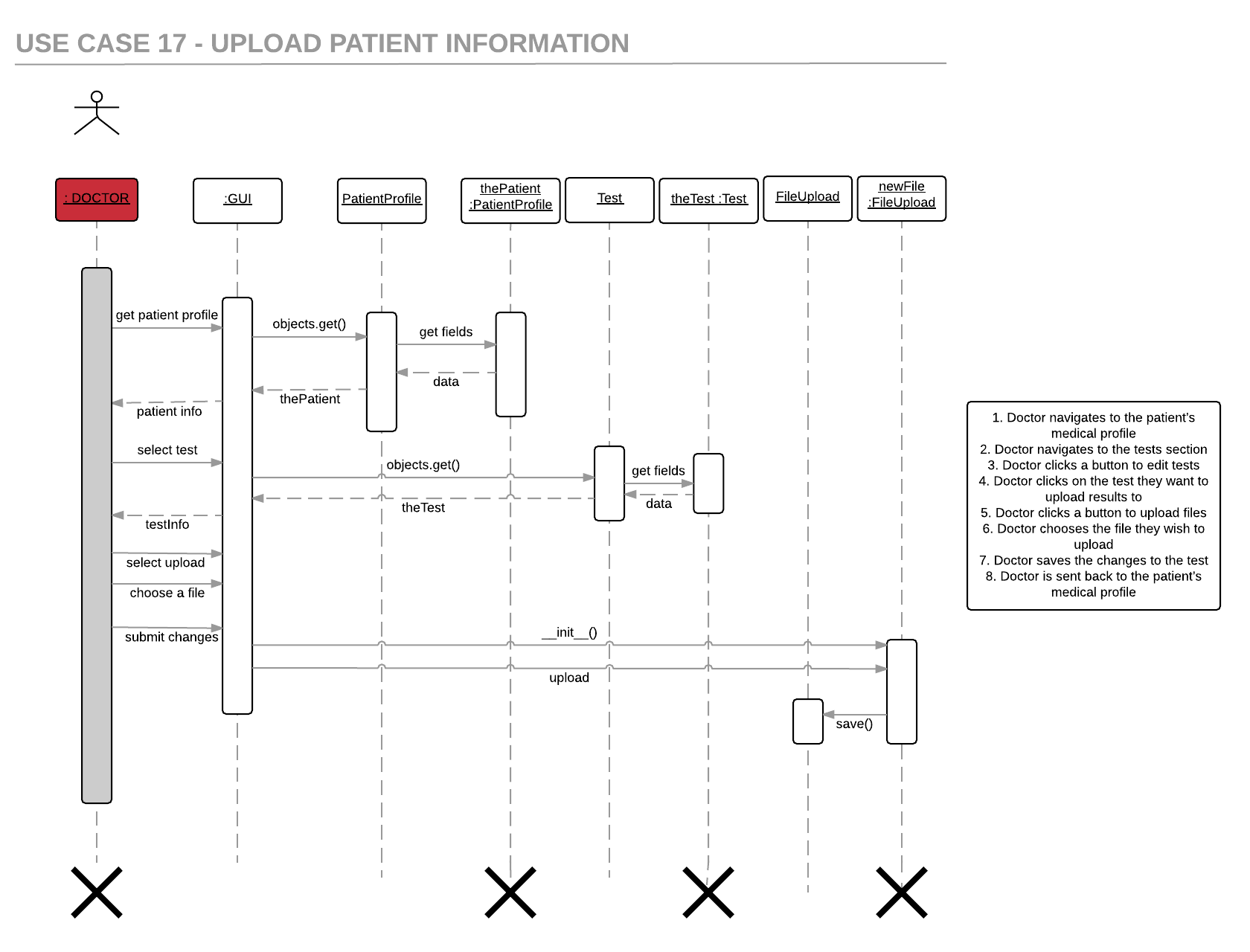












# 

# Design Rationale

**As of 2/22/16**

The premise of our project is that we have patient data stored in the database, and users need correct permissions to perform any operations on the data. The data is stored in dictionaries in the objects, which are pushed to the database. The permissions are implemented via an array of Booleans for each type of user, where their permissions are checked for any significant operations that they’re trying to perform.

Some minor design changes we’ve done so far:

Every user used to be extending a universal Person class. However, after deciding that admins had no overlap with nurses, patients, or doctors, we decided to make Admin a separate class.

Appointments used to be stored in Doctor. The rationale was that it would be easier for nurses and the calendar to iterate through them to view all appointments. However, it would be a convoluted process for a Patient to view their appointments (they’d have to go to their doctor object, get all their appointments, and then iterate through all appointments to find their appointments). To make it simpler, appointments are now stored in Person, and there will be a static getAllAppointments function that accesses the database directly.

We planned to create a logging class, to keep track of system activities. However, since Django has a built in logging api, we are putting all logging functionality into Admin. Function calls in Admin will act as the middle grounds between the project and Django’s api.

Other changes:

Decided on using an external calendar api.

Using a simple log model to track activities instead of django api, as learning the django api would take more time

There are no more explicit permissions, but what each user sees is restricted based on their profile type.

reworked base patient/doctor/nurse classes. They are all models now with formal relation keys in the database

Started using selenium for testing

Removed all direct add/edit functions for users, using views and forms to modify information instead

Decided on a general website style

Major revisions to site style and aesthetics