Product Design

Team **<Team 5 Tigers>**

Revision Number	*Revision Date*	*Summary of Changes*	*Author(s)*
0.0	02/19/16	Initial write-up	Robert Mason
0.1	02/23/16	Sequence Diagrams	Robert Mason
0.2	03/3/16	Fixing some of the sequence diagrams; the rest of these will need to be changed later. In addition, the whole class diagram was redrawn to make it better. This was accomplished by separating out each class into a separate diagram which are then stitched together to keep everything simple.	Robert Mason

Components and Functions

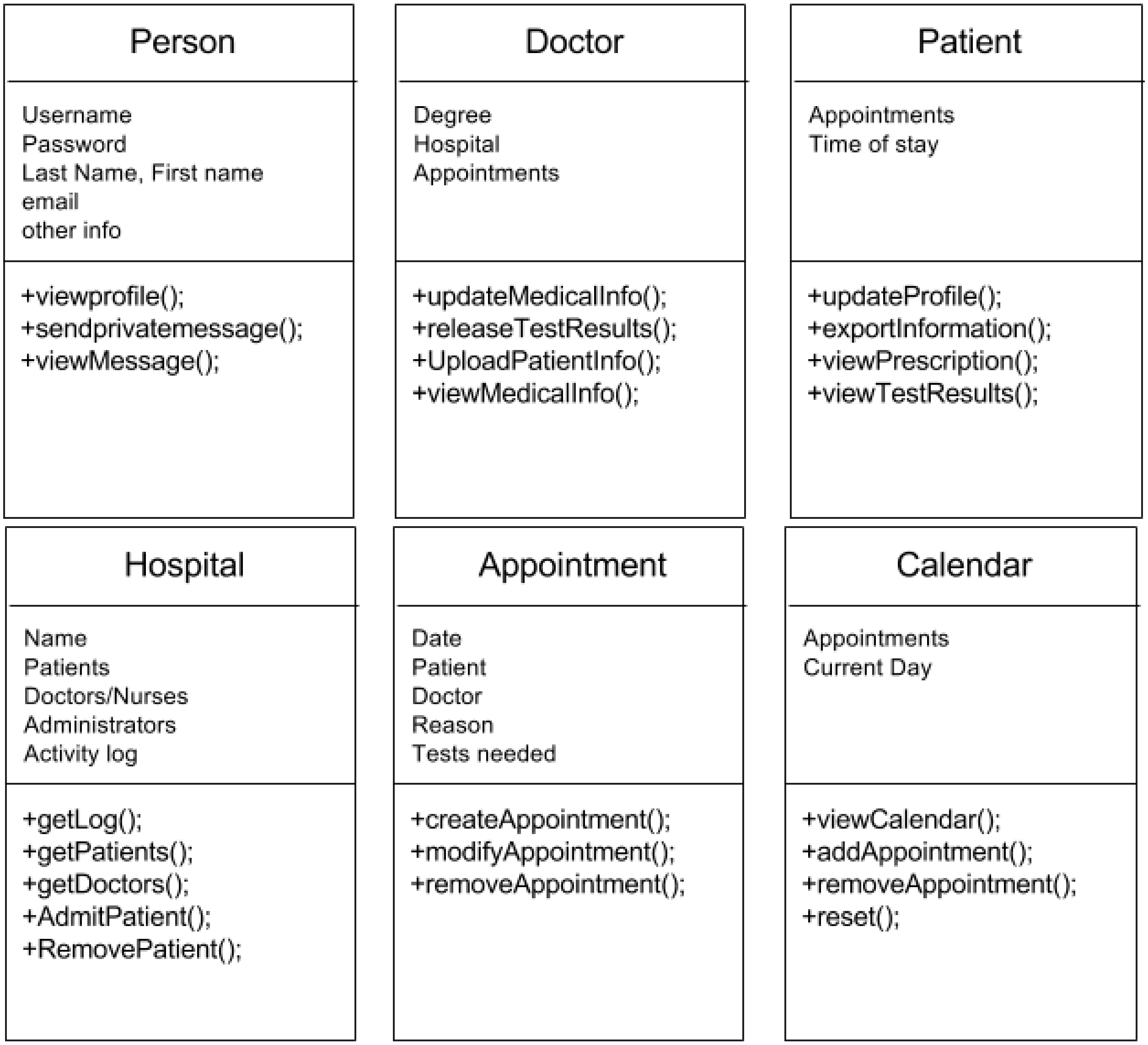
Contact	Component State: - Phone number(s), - Address - E-mail Component behavior: - viewMessage() - A person can view messages sent to him/he - sendPrivateMessage(Person) - A person can send a private message to another person
Prescription and test results	Component state: - Medicine name - quantity - dosage - lab test results Component behavior

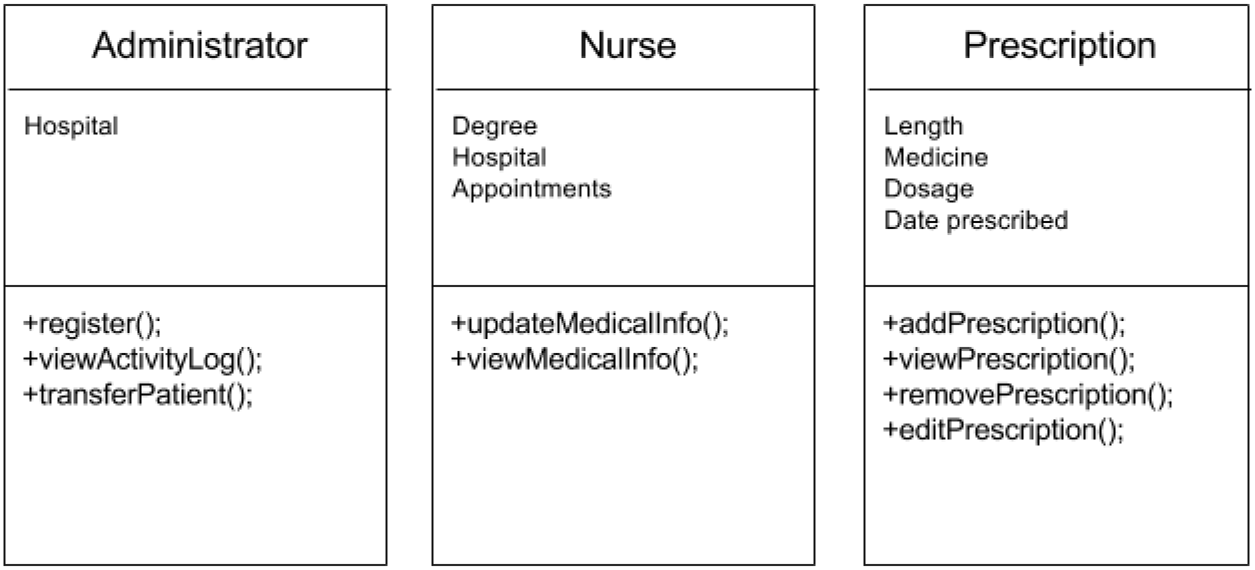
	- addPrescription(Prescription, patient) - doctors can add a prescription to a patient’s record - removePrescription(Prescription, patient) - doctors can remove a prescription from a patient’s record
Appointments	Component state: - Appointments scheduled - Appointments available Component behavior: - createAppointment(Patient, date, time) - Doctors, nurses, and patients can create appointments - updateAppointment(Patient, date, time) - Doctors, nurses, and patients can update existing appointments - viewAppointments(Person) - Doctors and patients can view all of their existing appointments, nurses can view all appointments for the day and week - cancelAppointment(Person, date, time) - Doctors and patients can cancel existing appointments
login/signup	Component state: login -user ID(patient & doctor) -user password (patient & doctor) signup - username - new password - email address Component behavior: - enterUsername() - people create a unique username for themselves - enterPassword() - people create a unique password to verify that they are the same person as the username - verification() - people enter their login information to verify that they are who they say they are - enterEmail() - people enter their e-mail address
Medical/profile information	Component State: - Name of person (first and last) - medical history - other physical properties (such as height or weight) - insurance information Component behavior: - updateProfile() - patients can update their profile information

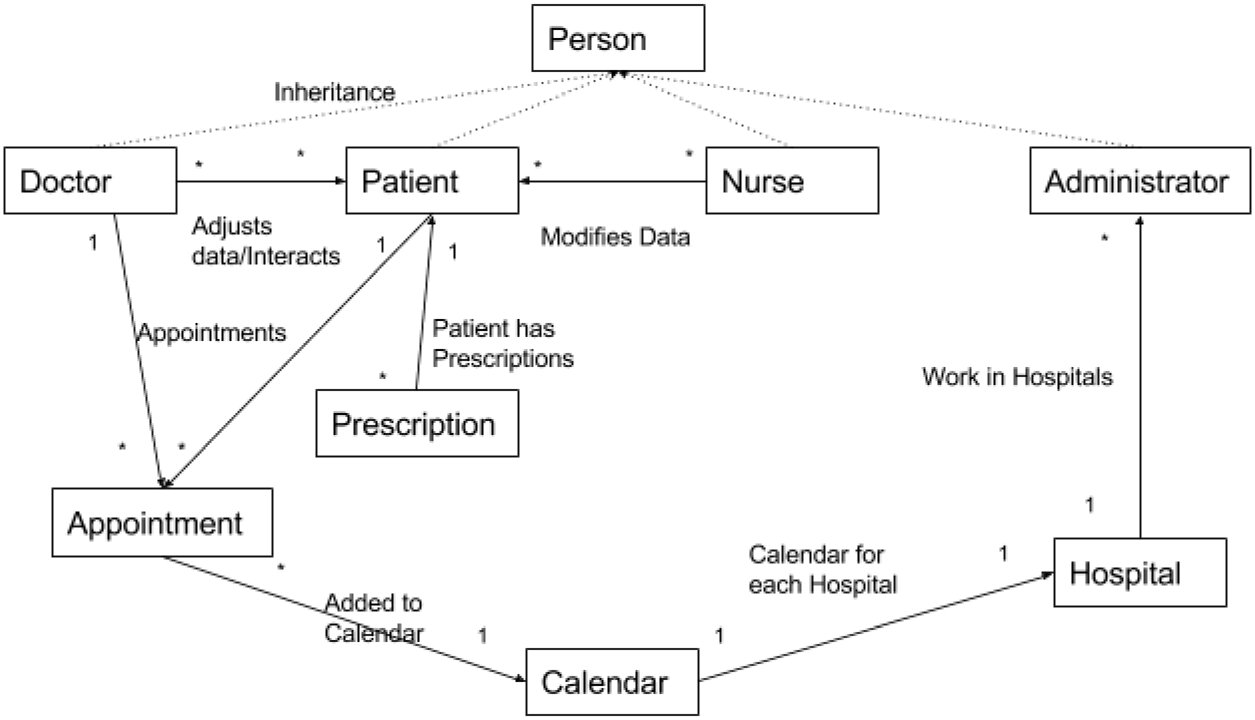
	- updateMedical(Patient) - doctors and nurses can update the given patient's profile information - exportInfo() - patients can export their profile information and test results with relevant privacy warnings

Class Diagram(s)

Class Diagrams are broken up into two separate sections in order to make it more viewable.

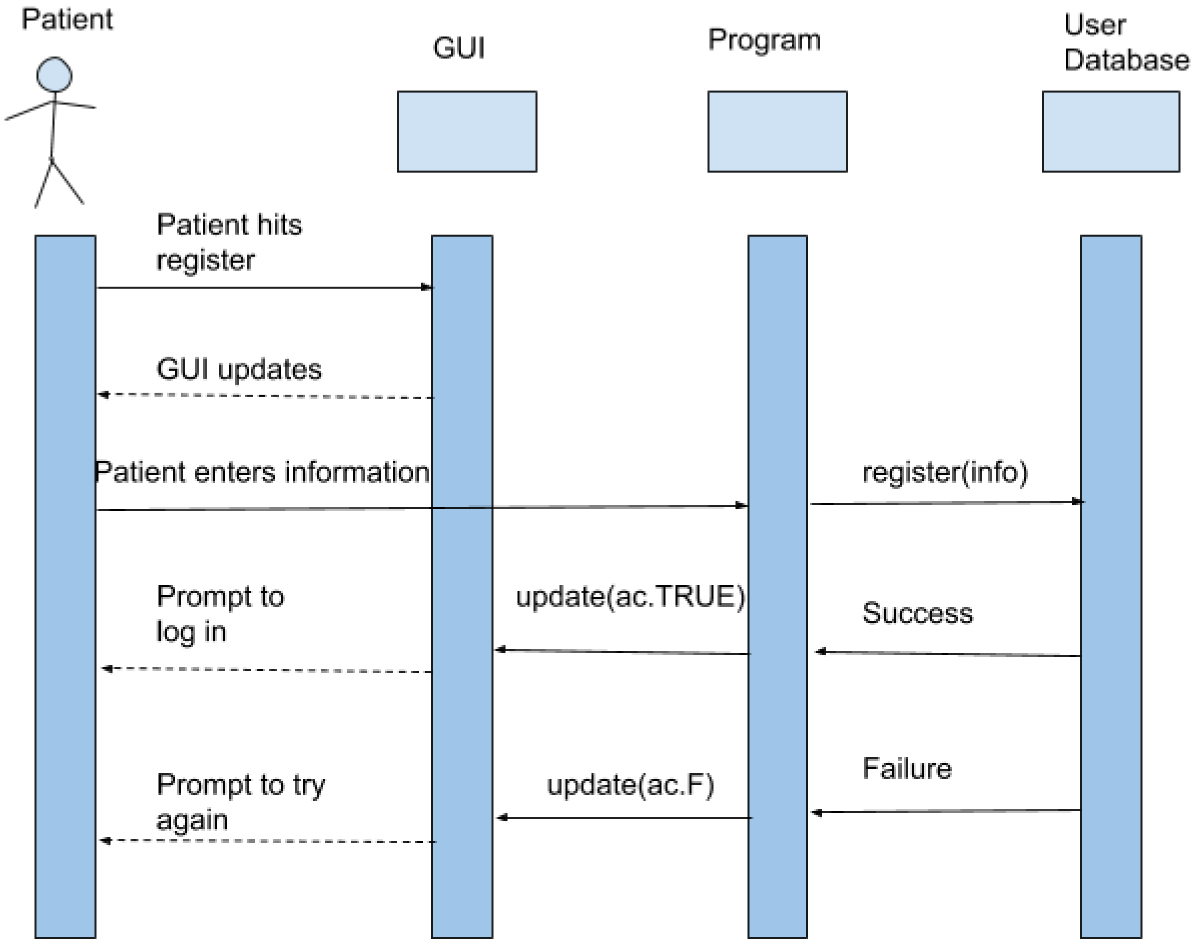




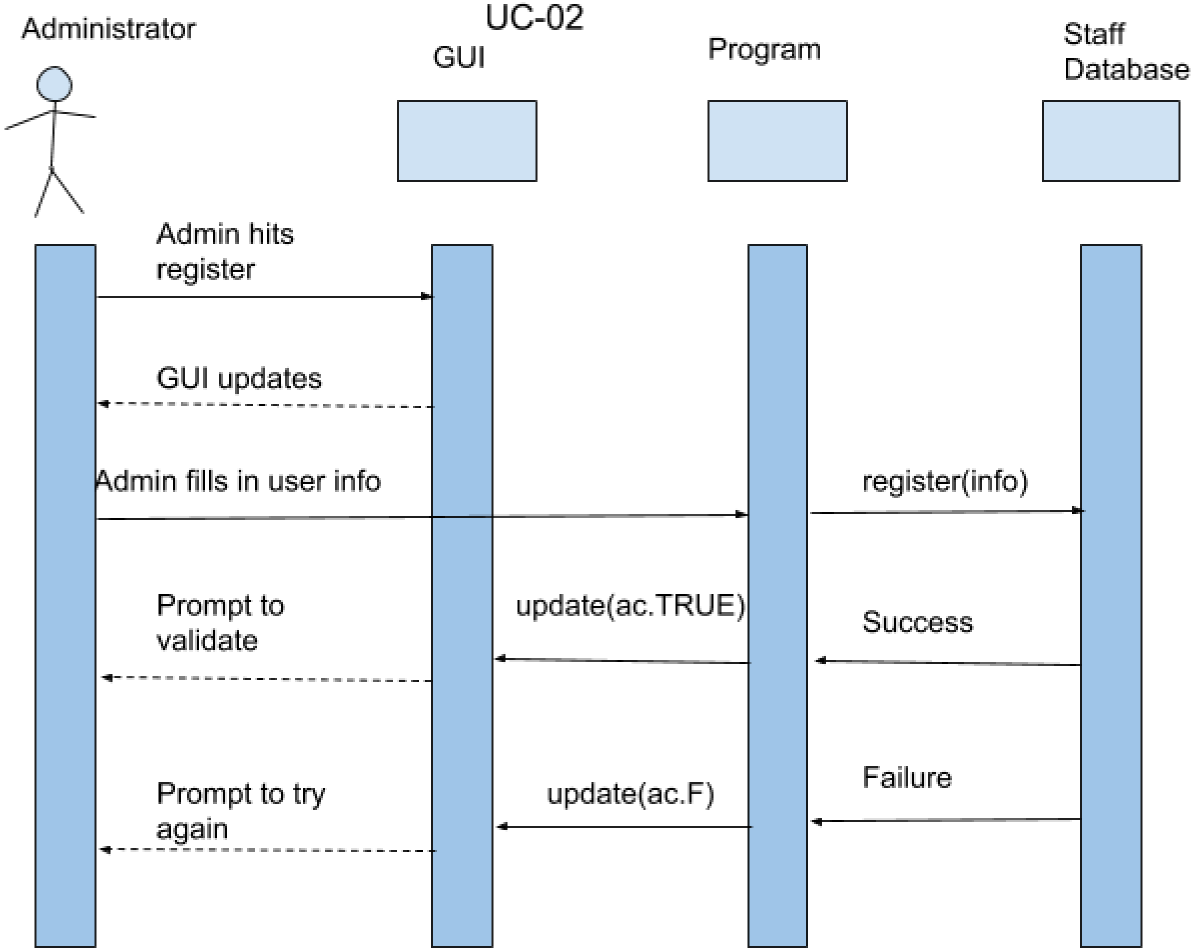


Sequence Diagram(s)

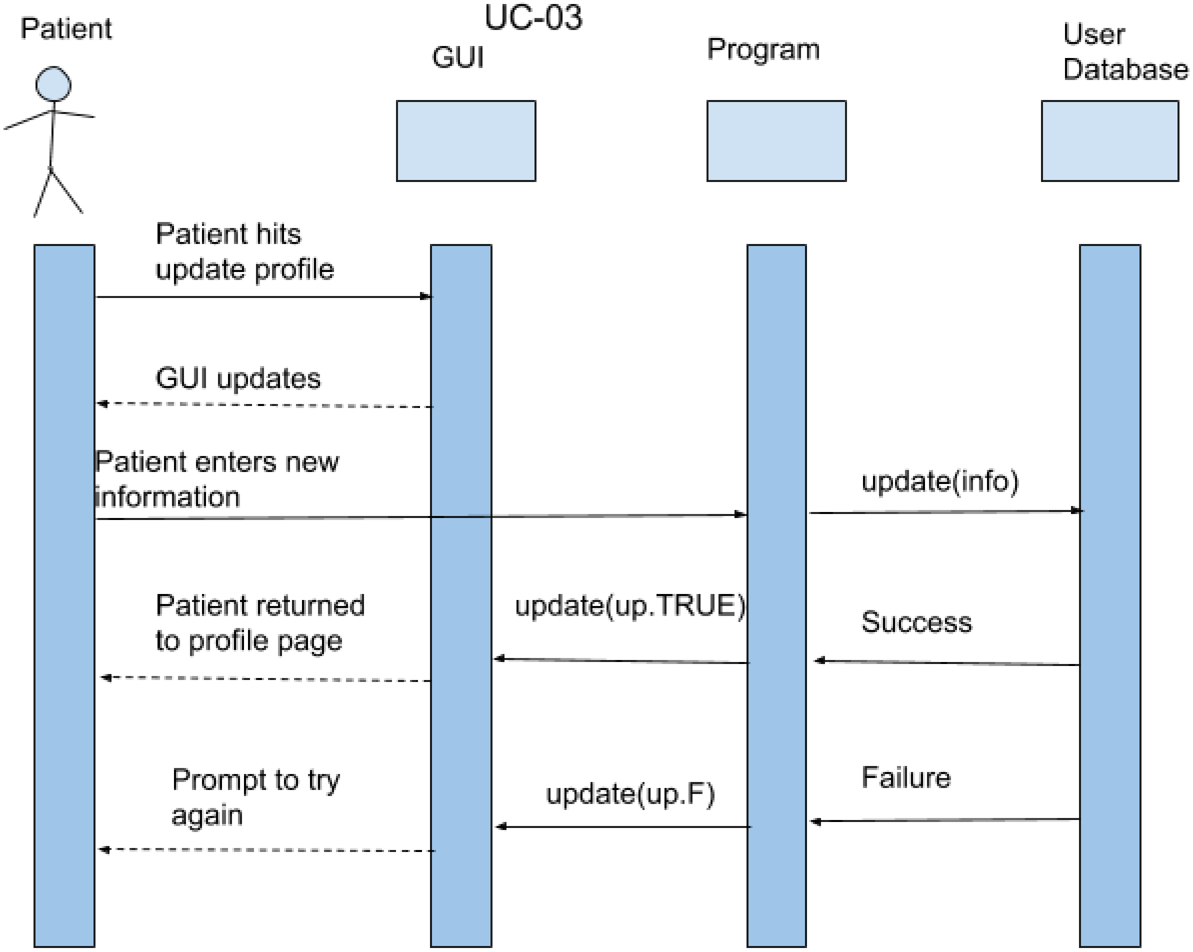
User Registration
UC-01



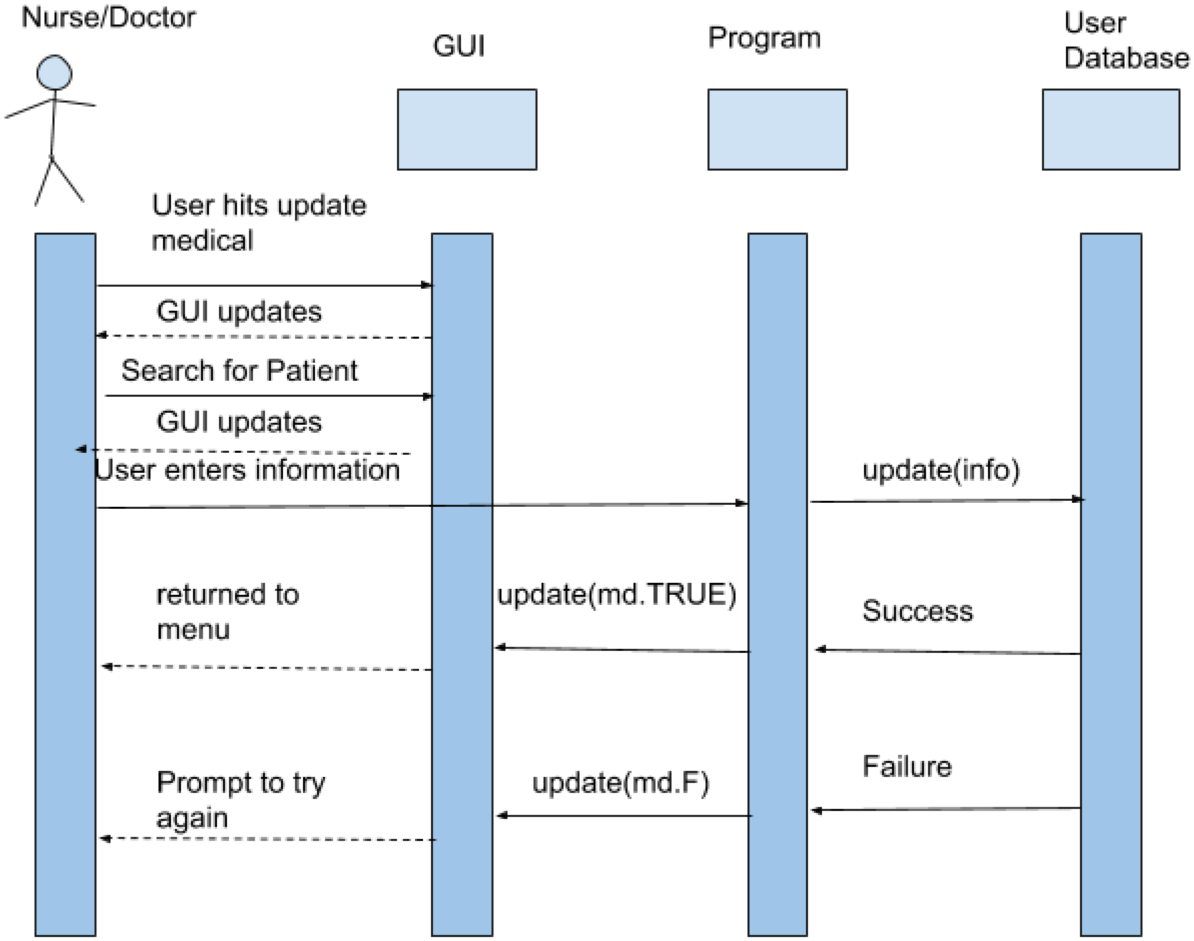
Administrator Registration UC-02



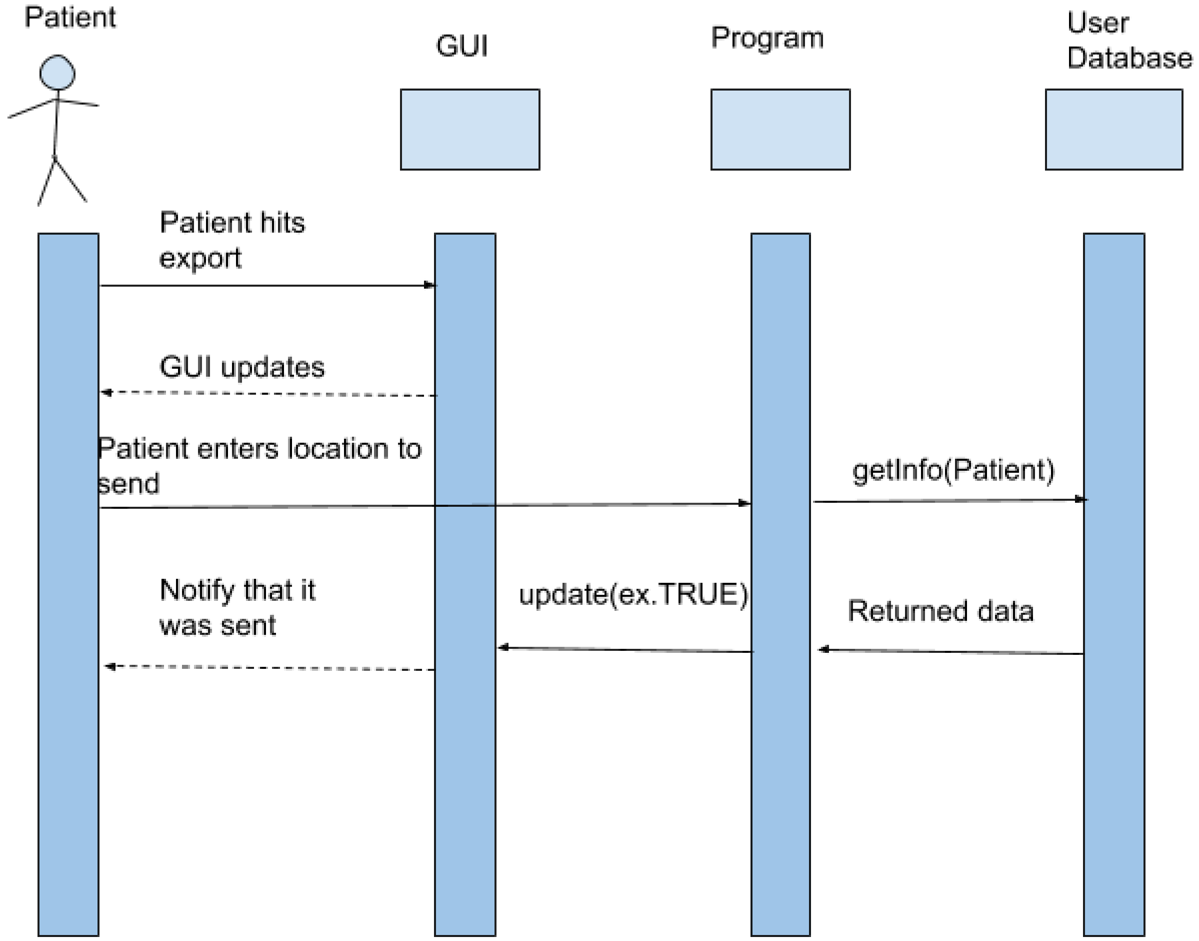
Update Patient
Profile
UC-03



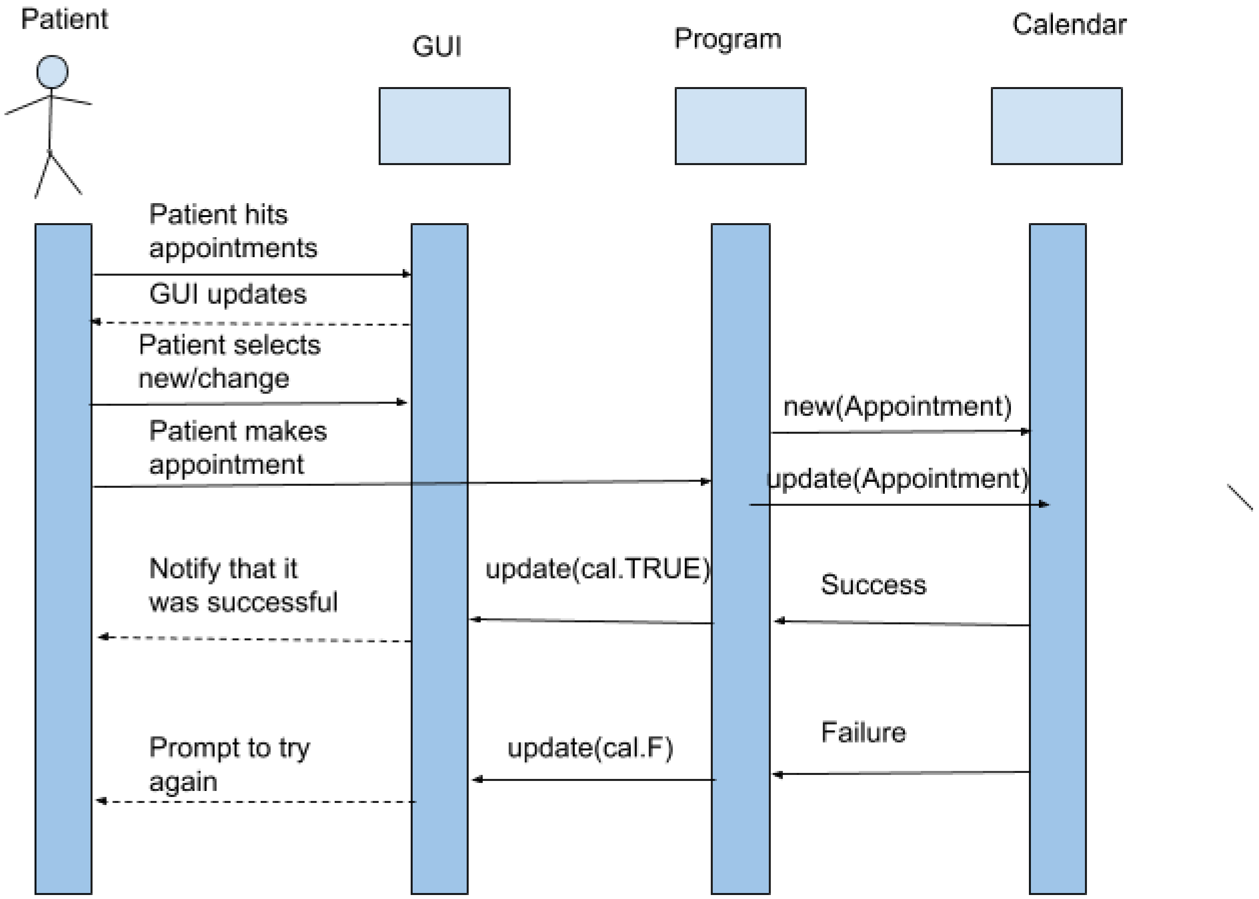
Update Patient Medical Information
UC-04



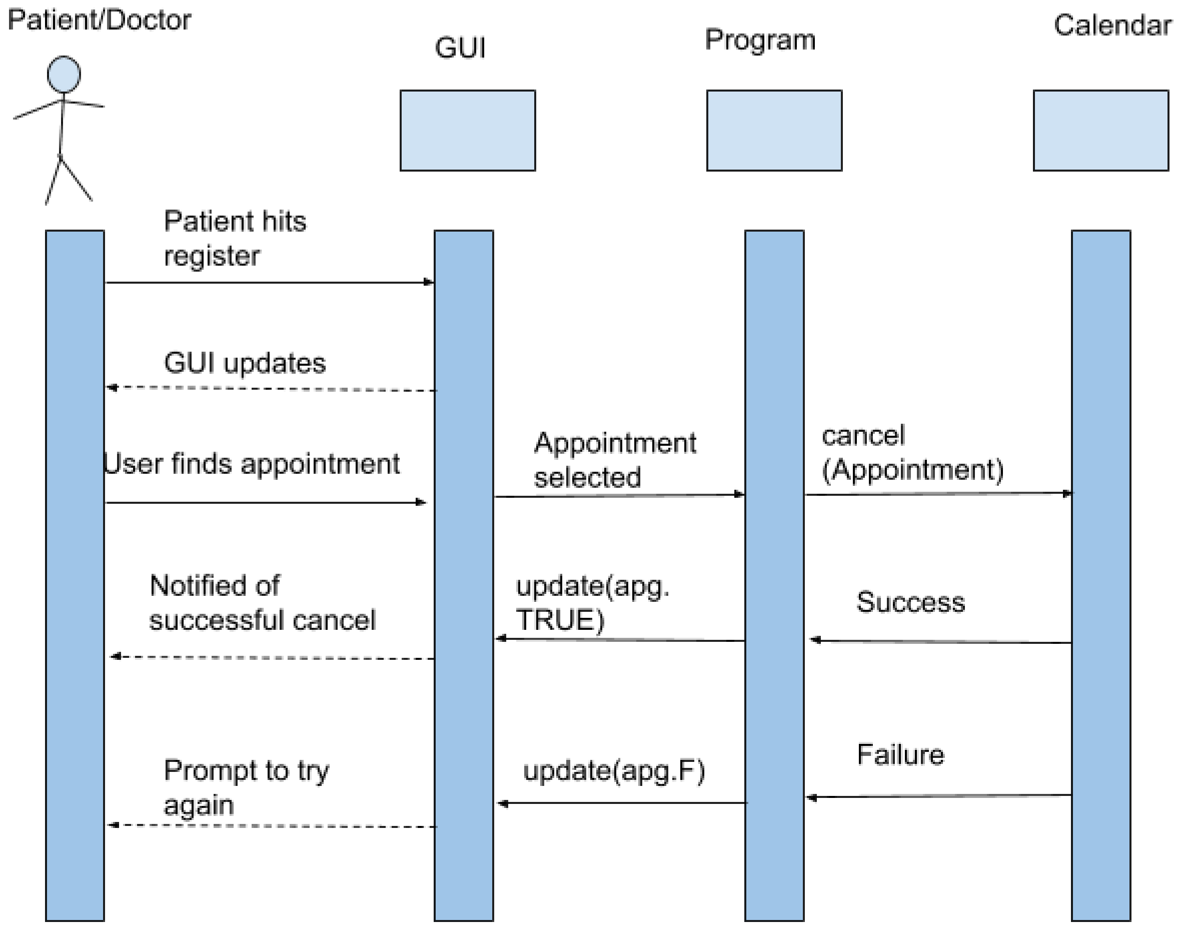
Export Information
UC-05



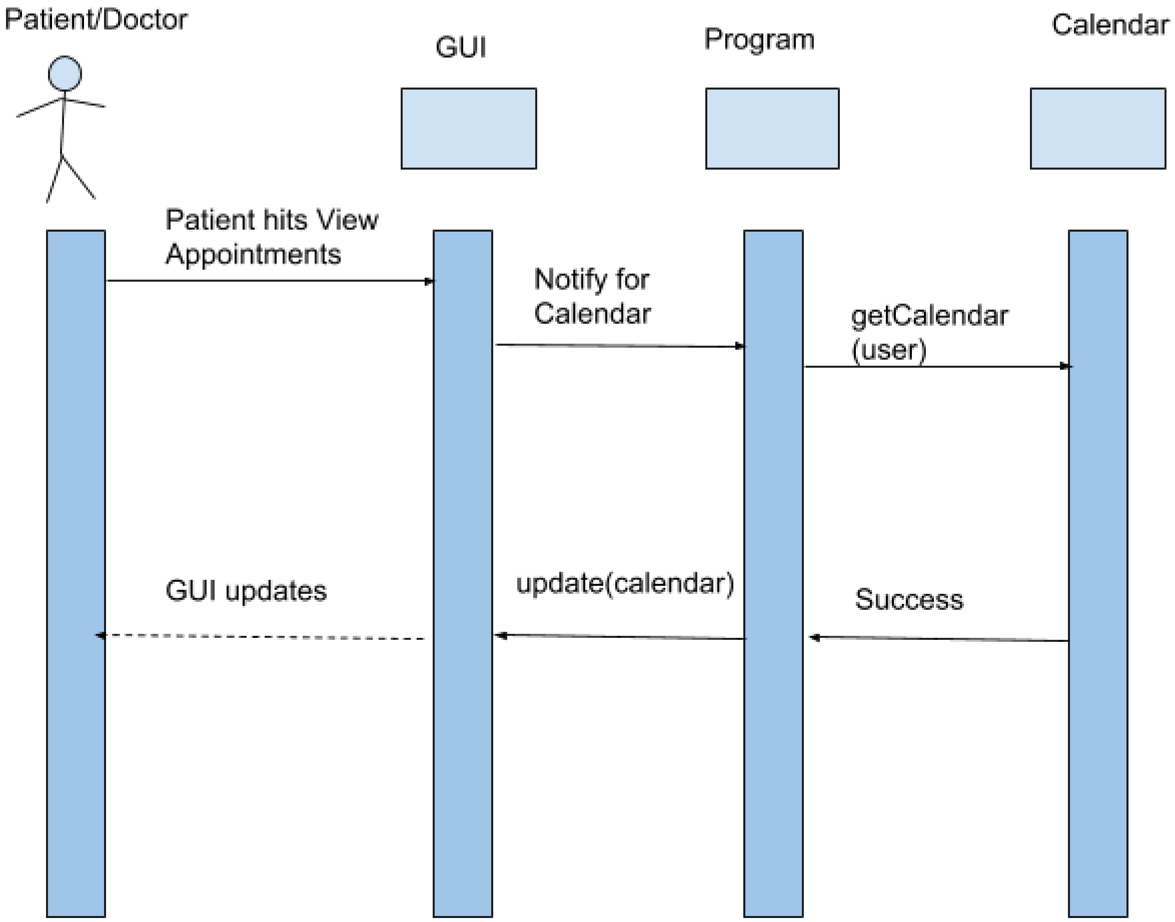
Create/Update Appointment
UC-06



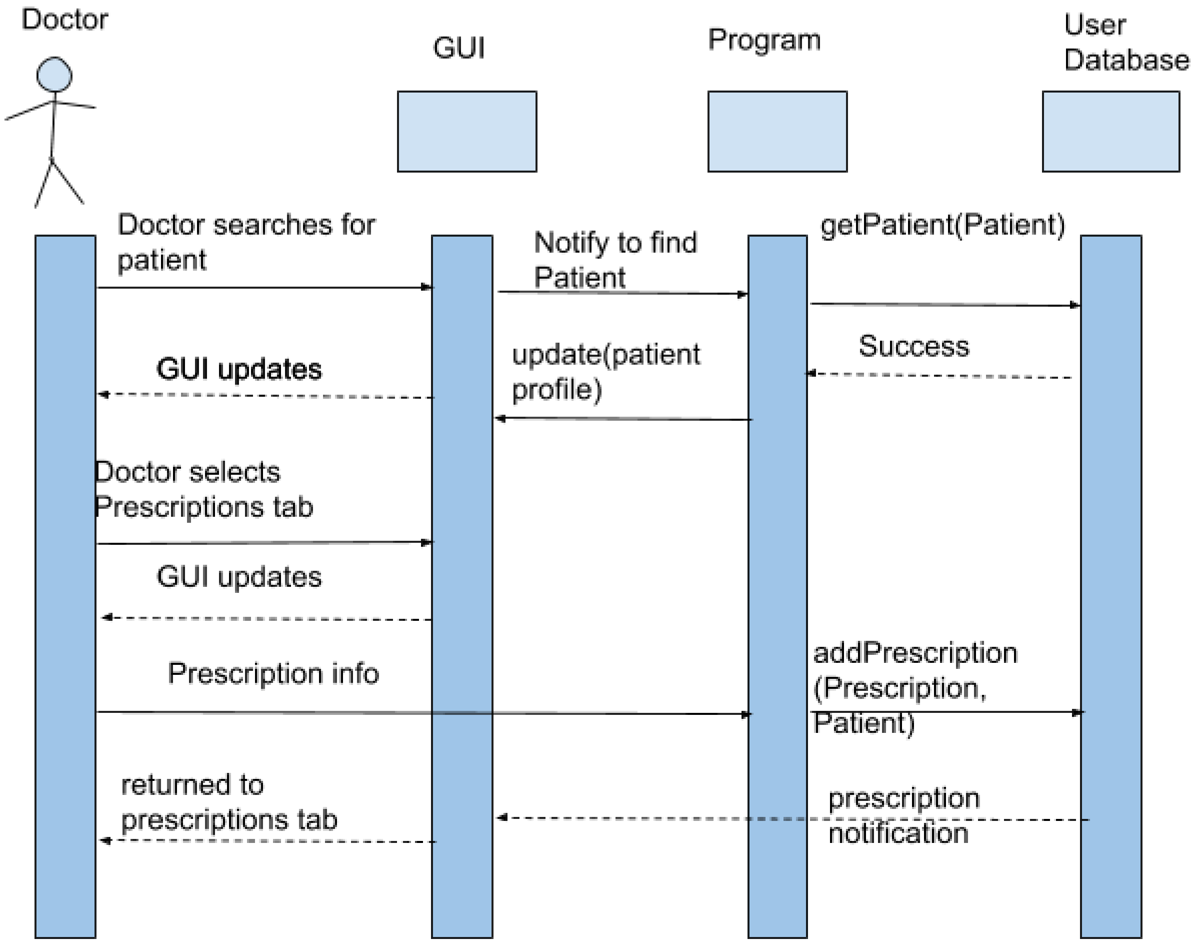
Cancel Appointment
UC-07



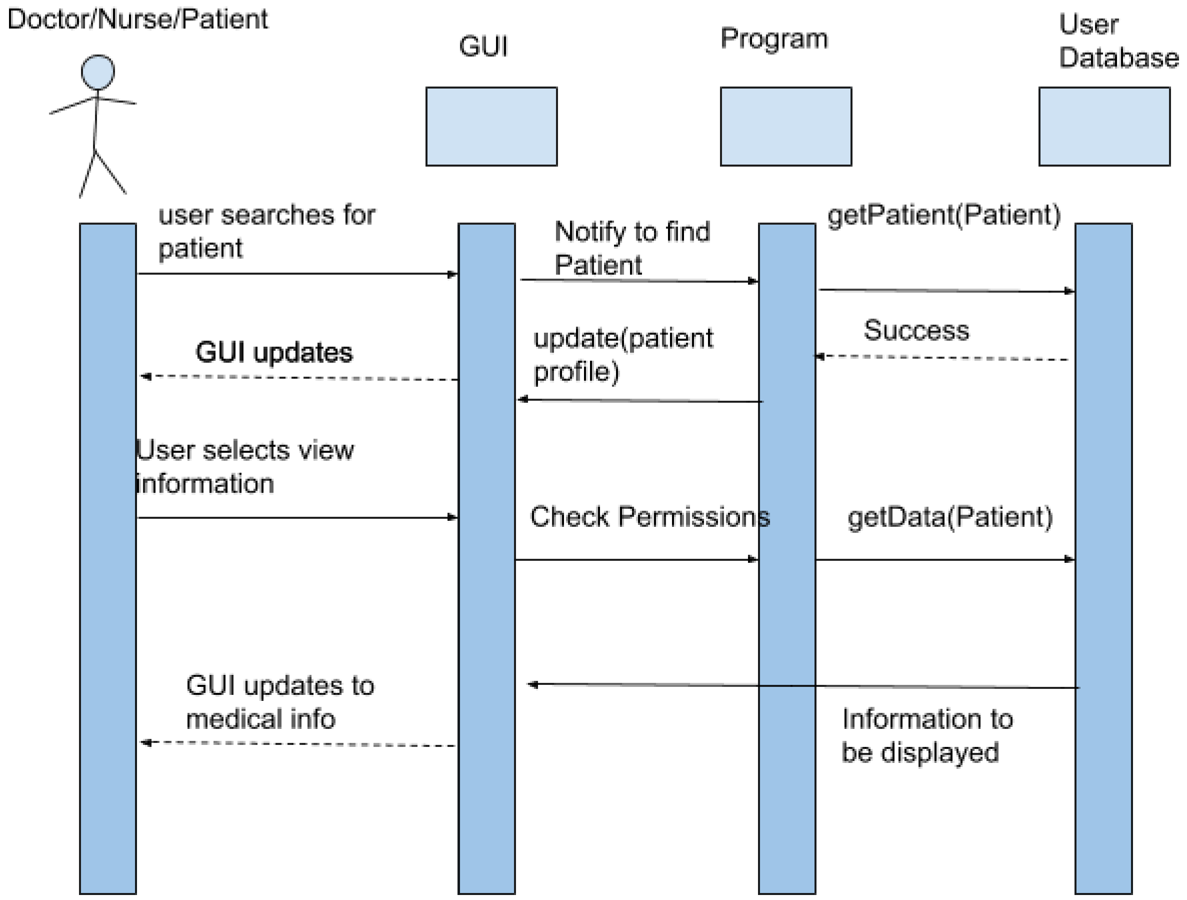
Appointment Calendar
UC-08



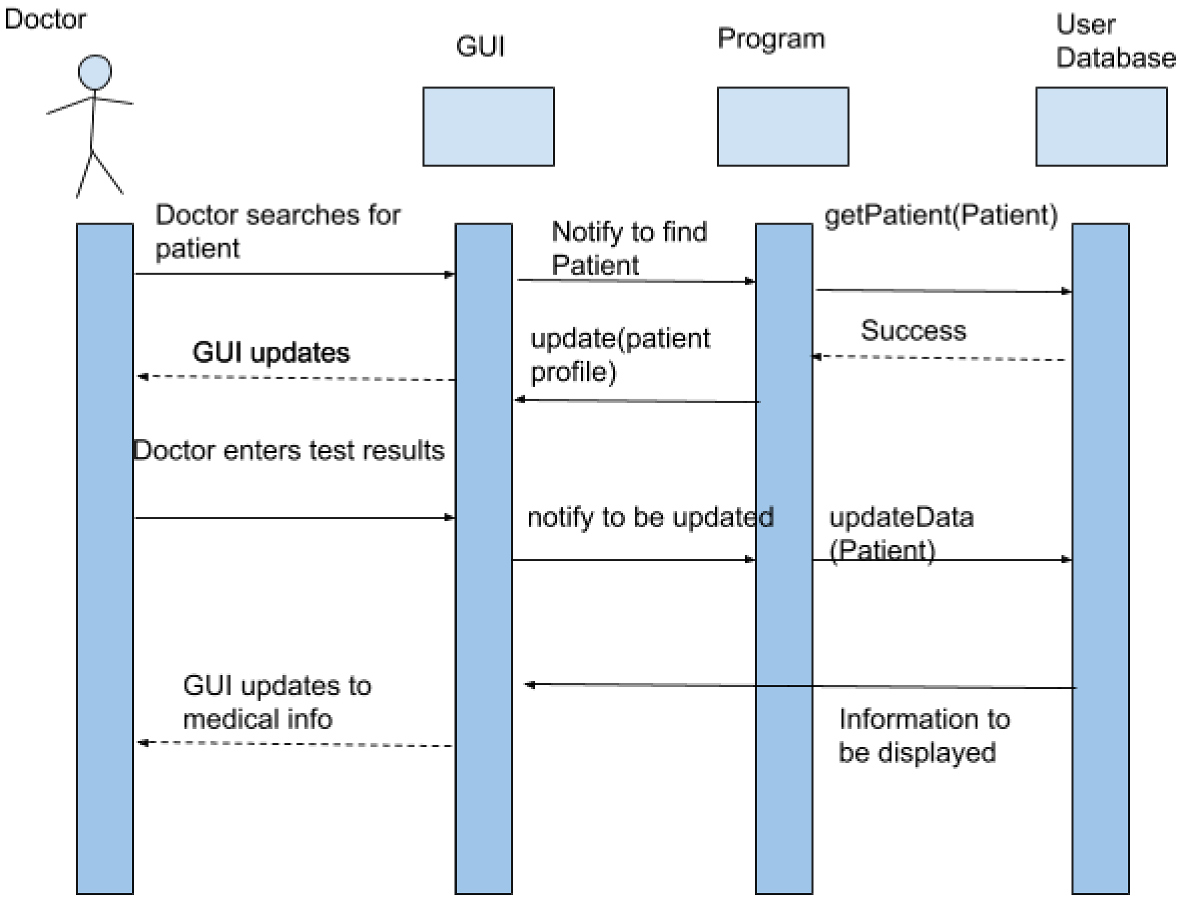
Add/Remove Prescription
UC-09


Doctor
GUI
Program
User Database
Doctor searches for patient
Notify to find Patient
getPatient(Patient)
Success
update(patient profile)
GUI updates
Doctor selects Prescriptions tab
GUI updates
Prescription info
addPrescription (Prescription, Patient)
prescription notification
returned to prescriptions tab

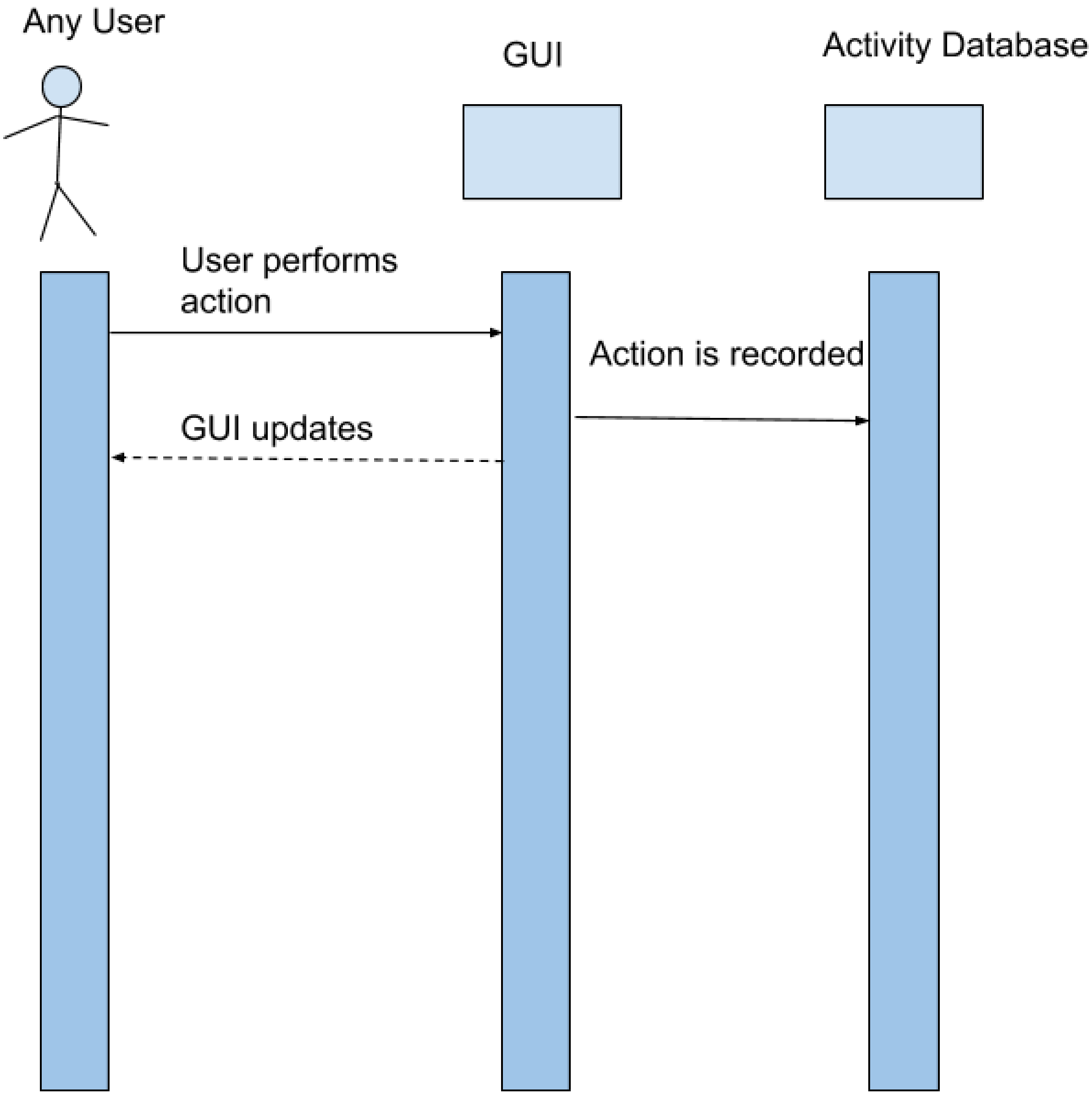
Viewing Patient info
UC-10



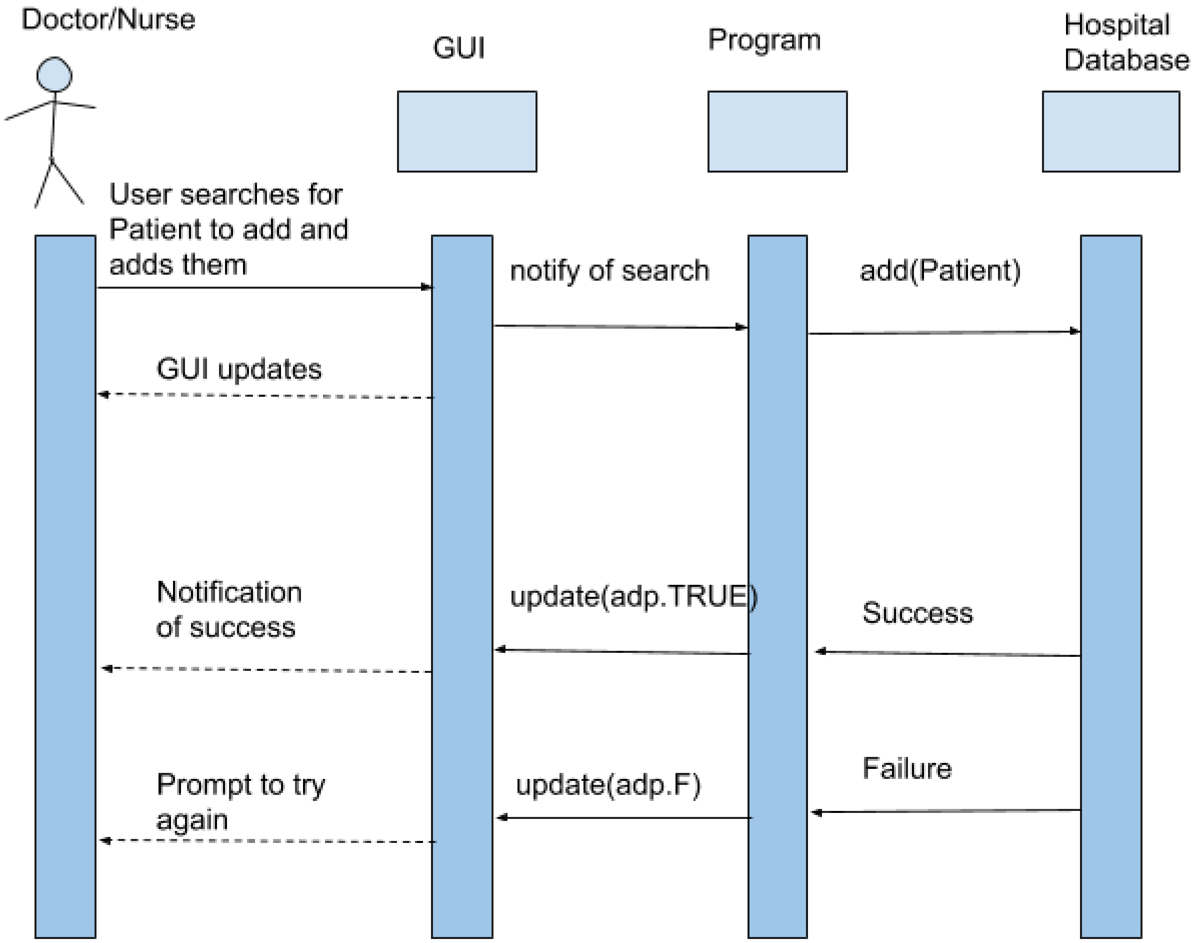
Release Test Results
UC-11



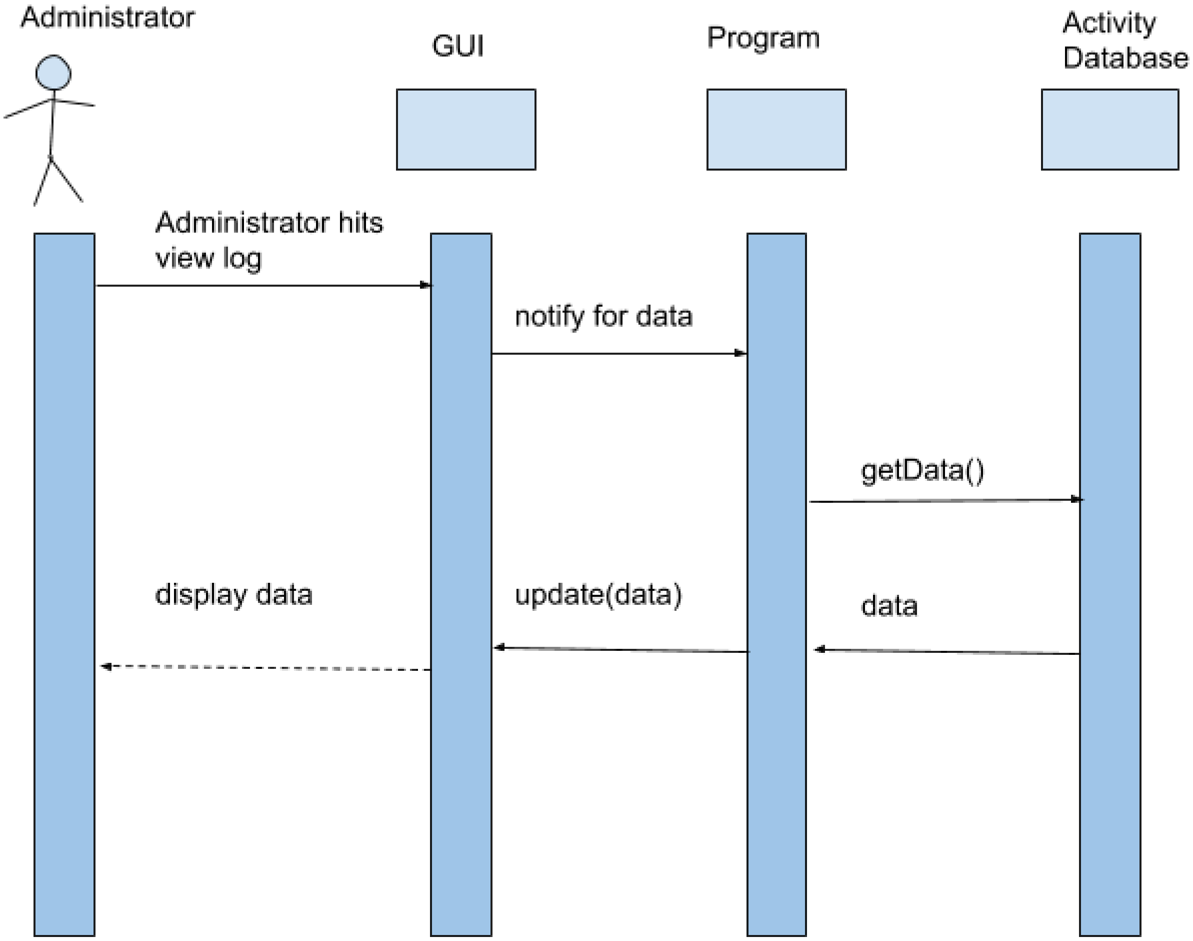
Logging System Information
UC-12



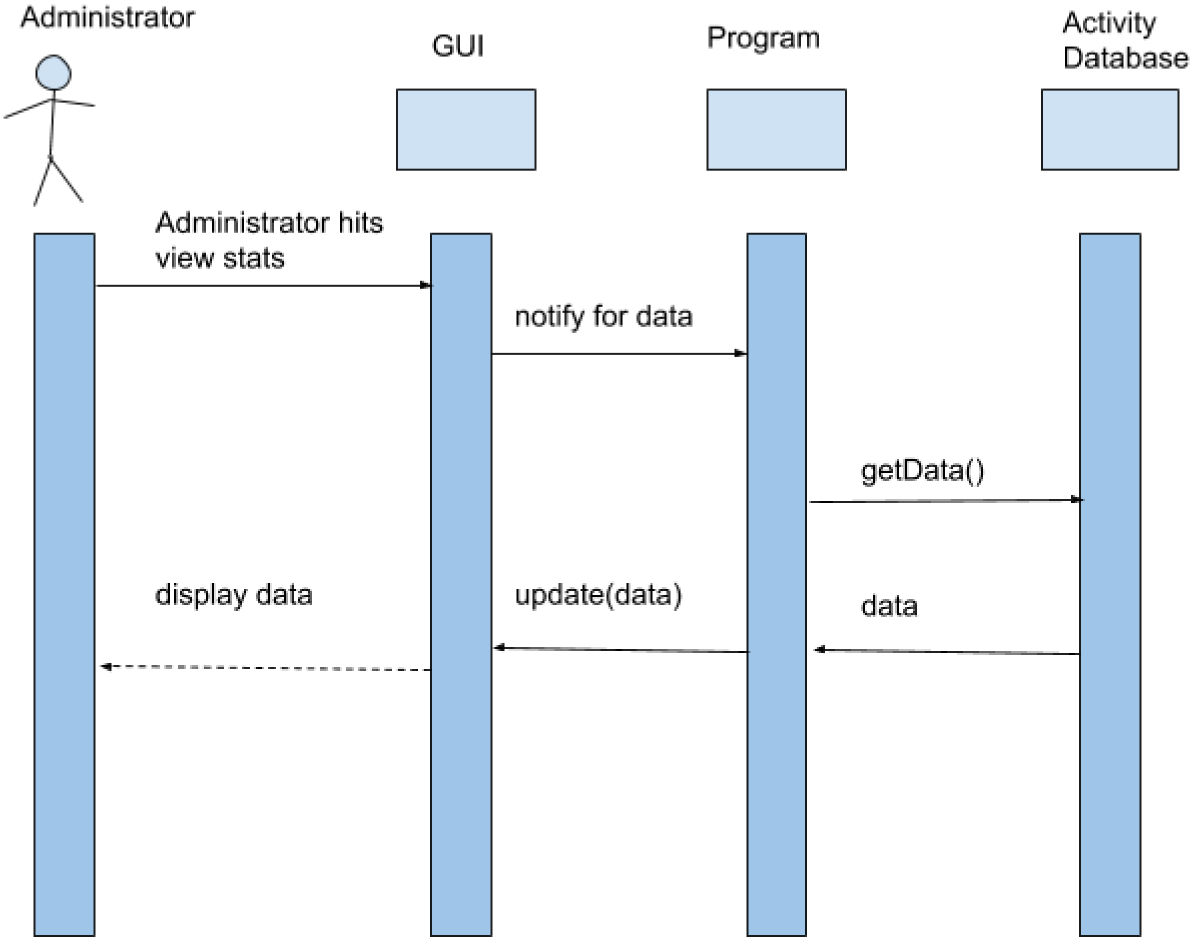
Admission/Discharge
UC-13



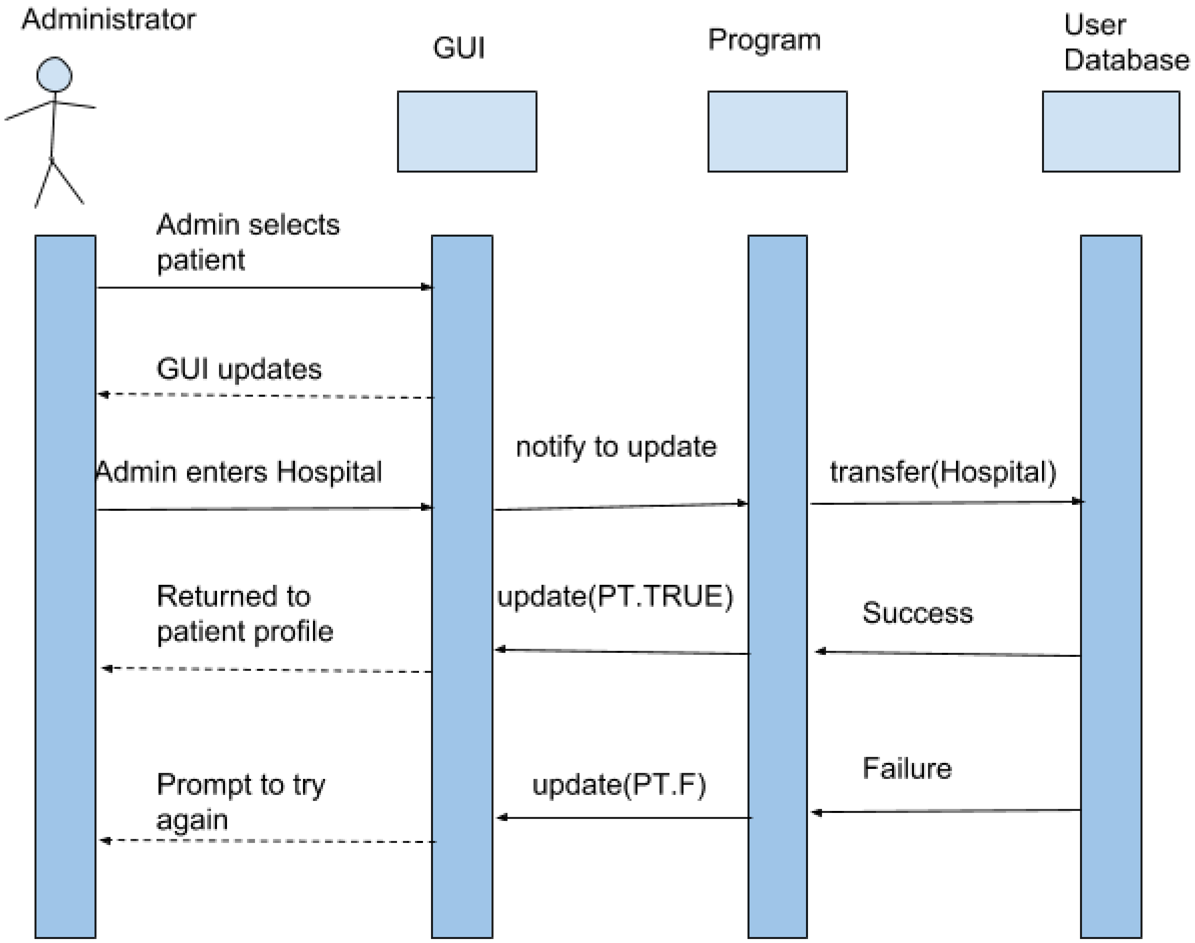
Viewing Activity Log
UC-14



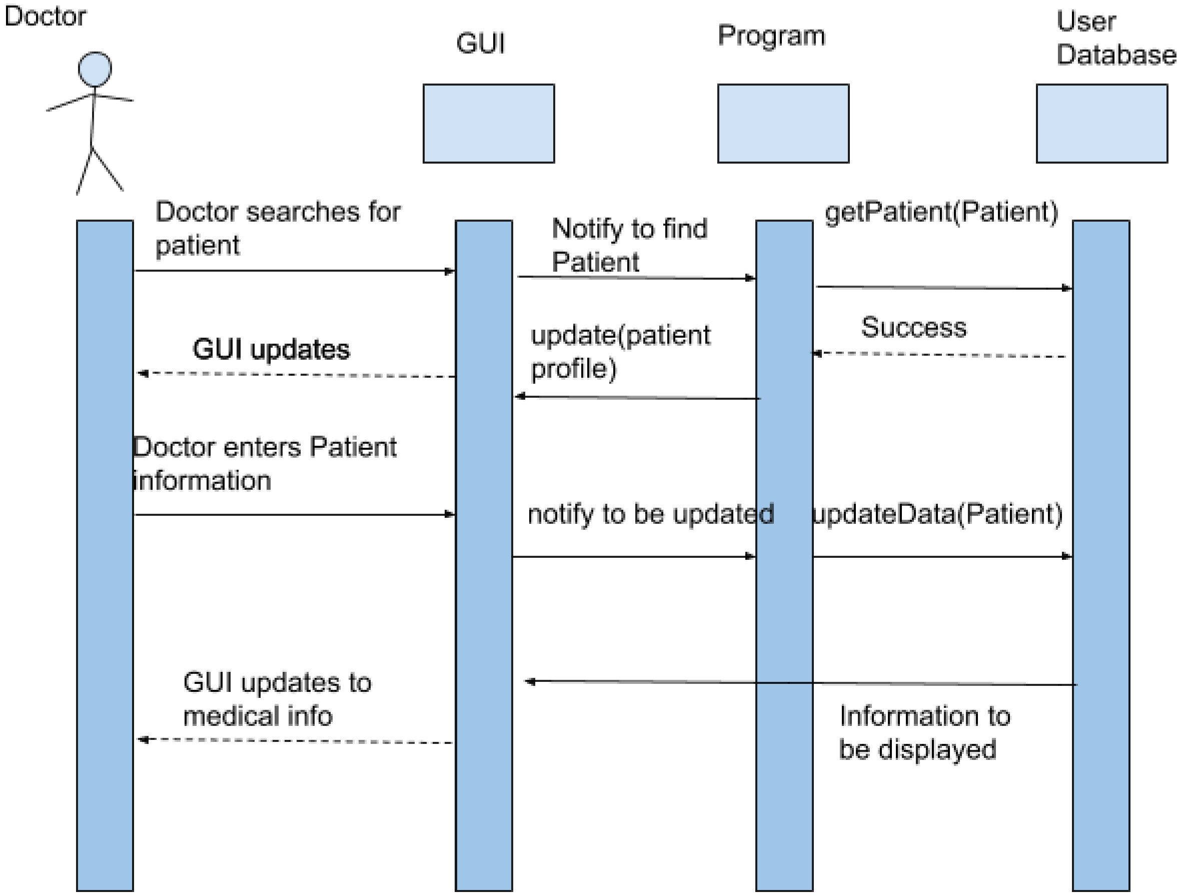
Viewing System Statistics
UC-15



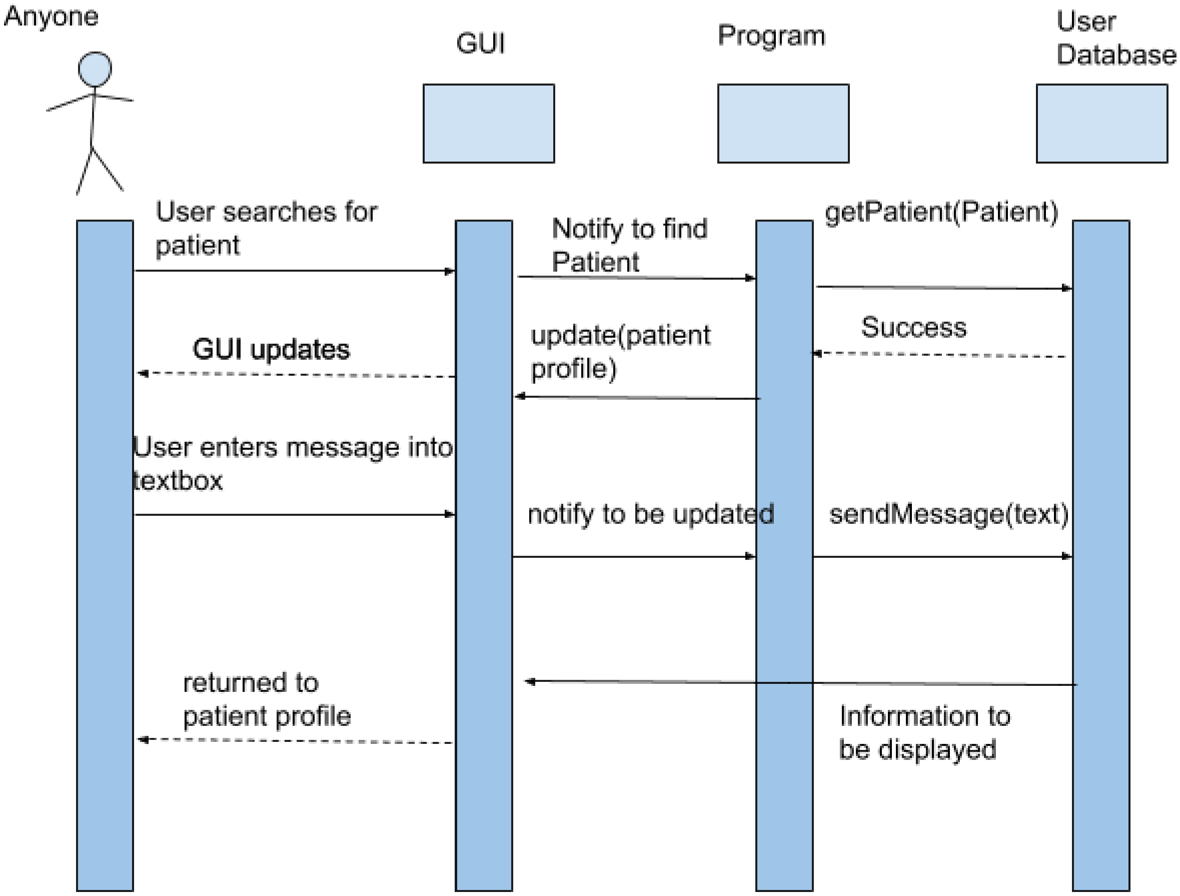
Patient Transfer
UC-16



Upload Patient Information
UC-17



Send Private Message
UC-18



Design Rationale

The biggest current issue that we have is how we are going to handle the prescriptions. There are a couple of possible ways of handling this but as of right now (2/19/16), we are going to use a system where each patient has a list of Prescription objects. This list can then be called by Doctors to be manipulated and a 'locked' version would be taken for if the nurse wished to view it. As far as the locking mechanism goes, it might just be that the method 'edit/modify Prescription' requires the person doing it to pass in their type, which would then be checked against what is allowed. We aren't sure though that that is the best way. Prescriptions could also be held globally so that any doctor has access to the list but nurses can only view it. The major concerns with this are that it is far too open and might become a memory hog in the long term.

Another potential concern in our design is how the various 'users' are separated and handled. This partially ties into the prescription section above because the main focus is on deciding how much data is stored within each user or should everything be separated out into various subsystems which would then be called when needed. As of now the way we are going to do this is by having a parent 'Person' class who would have daughter classes of 'Doctor', 'Patient', 'Administrator', and 'Nurse'. The super class would have very limited functionality that would only house methods that every user would have such as viewProfile and sendMessage. From there, each class would have get methods so that other users could pull information out. Similar to the Prescriptions paragraph above, the user requesting the data will probably have to submit their type as a parameter. The returned information (if any) would be 'null' if the requester had incorrect permissions. For methods in which no information is returned but is instead displayed, the returned boolean will just be false and nothing will be shown.

From there we can extend this to the Appointment Calendar. Each appointment is going to be it's own object that will be stored in a calendar. Each person will have an individual Calendar for use. We aren't entirely sure how that will be stored yet. It will either be that each user has a personal calendar or that there will be a global calendar that just specifies who each appointment applies to. As of right now it will probably just be with individual Calendars because it will be easy. The only issue is that data will have to be entered for each person who is involved in each appointment. In the future we might want to redesign this so that there is a master calendar for each hospital in which appointments are added and removed. This way information would only be entered once into the system. The only drawback is that it would make it more difficult for the system to find a specific appointment.