Albert Wallace -- aew0024
COMP3700
HW4
24 February 2013

[#1] Concept statement & sample use cases, with basic class diagram:

Medical information management system (MIMS) is a database management system that provides the capabilities for doctors, nurses, laboratory testers and management personnel to access patient records and provide services. MIMS interacts with other ancillary computers in the network system across the nation. Links are provided to existing computer systems, such as those used by the pharmacy, laboratory, and radiology departments. The medical staff communicates electronically with these systems from MIMS and views appropriate actions taken on their behalf. For example, prescriptions generated in MIMS are instantly time-stamped and routed to the pharmacy computer system for dispensing. When the order is processed, the pharmacy computer system routes the information back to MIMS and the patient's electronic chart is updated. In this manner, MIMS provides seamless communication between health care providers and ancillary service areas across the network system.

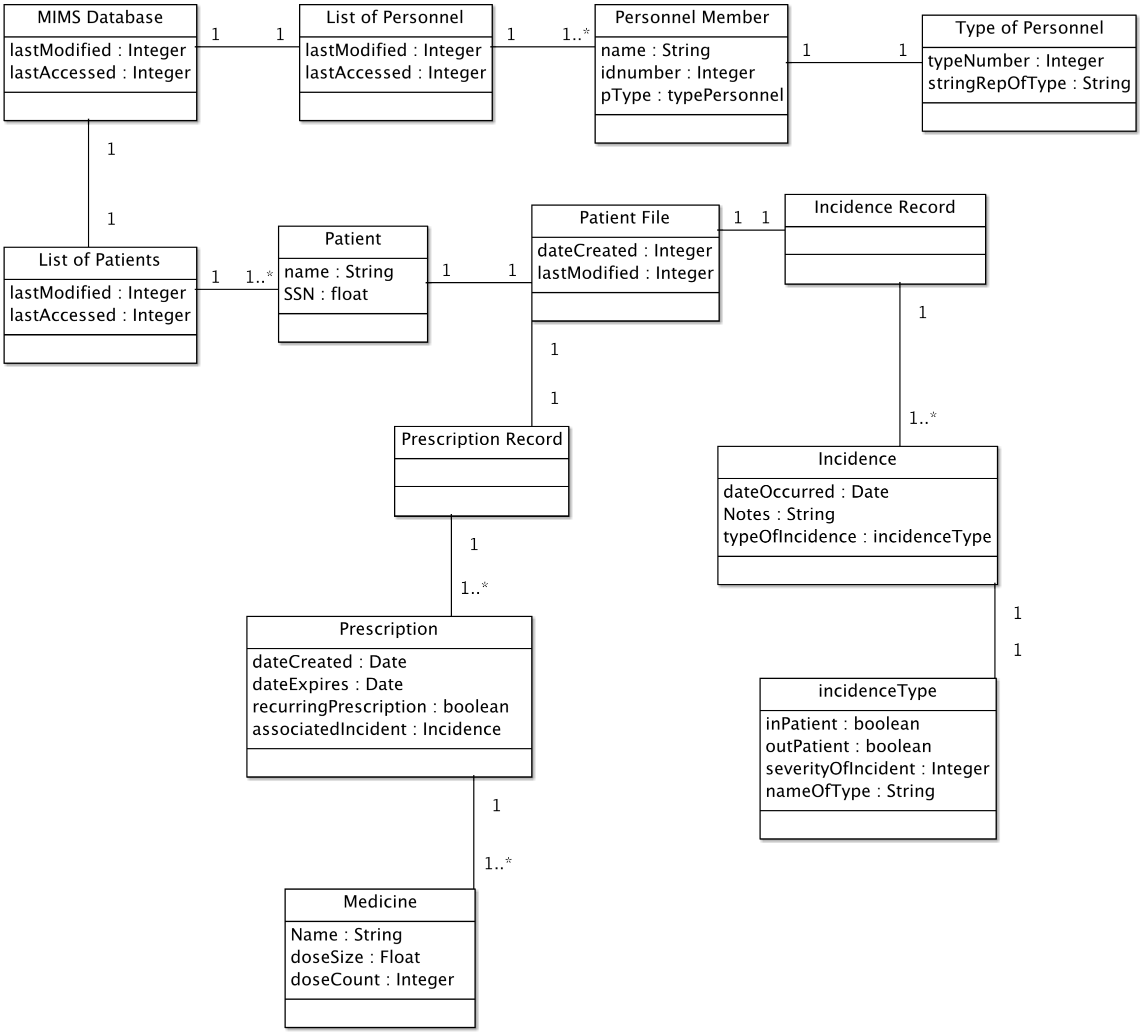
The system will support creation of a new patient, editing of current patient information, creation of prescription data for patients (writing prescriptions), checking prescription information, creation of new events in the patient's history ("illnesses"), display of past events in the patient's history, creation and display of non-emergent events ("other events") in the patient's history, and whatever else is required as specific requirements for the system are uncovered. Patient information will be linked to a central MIMS database, allowing access from multiple computers within a location, across multiple locations.

A sample of the use cases is as follows:

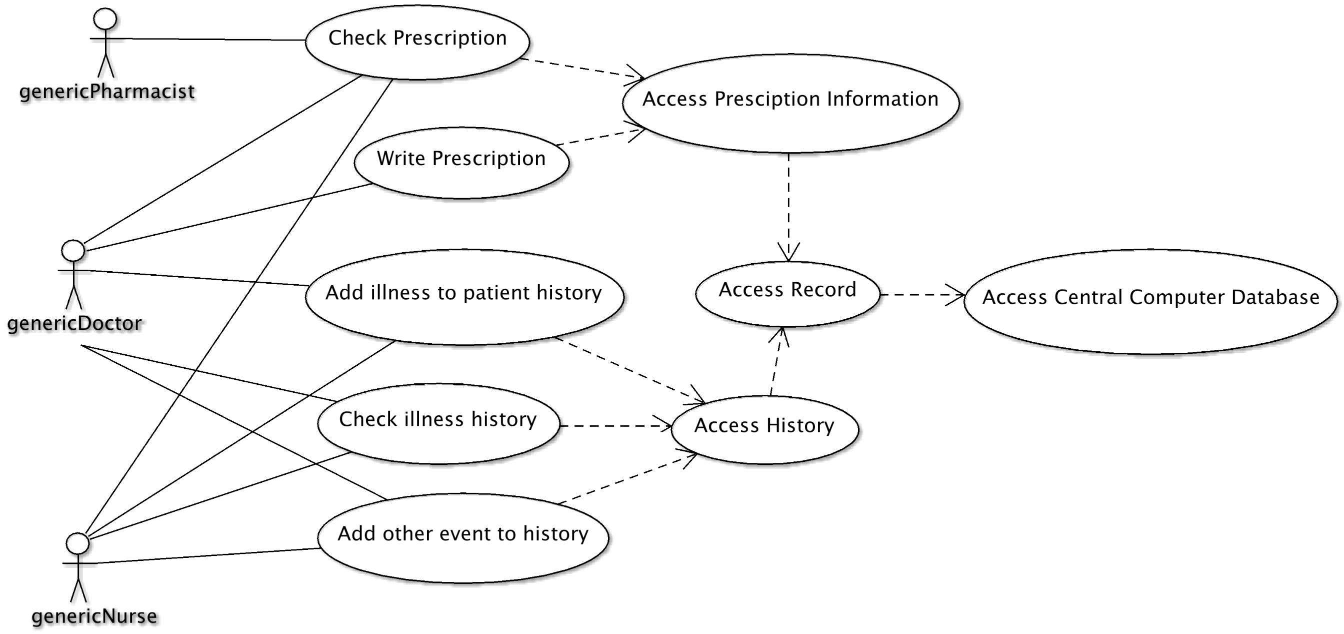
- Check Prescription:
  - In this situation, the pharmacist or doctor will attempt to log into the system (if not already logged in), then will request the recent history of a given patient's list of prescriptions. With this request, the central computer will return the desired recent history, offering all necessary information for each prescription listed. Nurses and select other authorized medical personnel will be able to access this data.
- Write Prescription:
  - In this situation, the doctor will attempt to log into the system (if not already logged in), then will access the patient's prescription history to add a new prescription. Once the fields are filled, the doctor can save the new prescription or cancel it. If the doctor chooses to save the prescription, the prescription will be written to the patient's file on the central server.
- Add illness to patient history:
  - Should a patient be sick, the doctor will log into the system (if not already logged in), open up the patient history, and select an option to fill a designated form regarding the illness, including name of illness, relative severity, date of occurrence, and recommended treatment.

Once the form is filled out, the doctor will submit it, saving the data to the patient’s file on the central server.

- Check illness history:
  - Should a patient be in a waiting room or need services rendered, the doctor will log into the system (if not already logged in) and access the database to check on the history for a given patient. The doctor will be able to check all details associated with a given bout of an illness for a patient. Nurses, lab testers, and select other authorized medical personnel will be able to access this data.
- Add other event to history:
  - In the event that a doctor will need to add an event (such as a typical checkup) to the history for a patient, the doctor (or other authorized medical personnel) will log into the system, fill out the form for the event (including name of event, type of event, date of occurrence, place of occurrence, and the like), and submit the data to the MIMS central server system for storage.
- Create new patient
  - In this use case (not pictured, but anticipated), a doctor, nurse, or other qualified medical personnel member logs into the system (if not already logged in) and fills out a form to create a new patient record in the database for a new patient. After filling out the form, the doctor or other authorized professional submits the form, and the information is appended to a new, blank database record in the MIMS database.



## [#2] Sample basic use-case diagram



[#3] Sample detailed text descriptions of system/actor sequences

--Check Prescription

1. Doctor (or other qualified medical professional) attempts to log into the system, or re-establishes an old session.
2. The interface authenticates the medical professional.
3. The central server returns an authentication token, and the interface indicates a successful login, allowing the medical professional to proceed.
4. The medical professional searches for a patient by name &/or SSN, or other data unique to one person.
5. The interface passes the query on to the server, the server returns results, and the interface displays possible matches.
6. The medical professional selects the intended patient from the list.
7. Correct patient chosen, the interface presents a list of options. The medical professional chooses to check the prescription history for a given patient.
8. The interface passes this query along to the database, which then returns a number of present and past prescriptions. The user can request more results (if any exist), or exit from here.
9. Once the user is done viewing all the necessary history information, the user logs off (or the interface automatically times out and initiates a log off).
10. The session and connection to the database are both terminated.

--Write Prescription

1. Doctor (or other qualified medical professional) attempts to log into the system, or re-establishes an old session.
2. The interface authenticates the medical professional.
3. The central server returns an authentication token, and the interface indicates a successful login, allowing the medical professional to proceed.
4. The medical professional searches for a patient by name &/or SSN, or other data unique to one person.
5. The interface passes the query on to the server, the server returns results, and the interface displays possible matches.
6. The medical professional selects the intended patient from the list.
7. With the correct patient chosen, the interface presents a list of options. The medical professional chooses to create a prescription for a given patient.
8. The interface presents a form to the user for the new prescription, including the prescription number, intended expiration date, verification of the current date, name of medicine(s) associated with prescription, dose count, and refill count, among any other proprietary information to be associated.
9. With the form filled out, the interface passes this data along to the database. The database will create a new prescription entry for the given user, then fill the blank prescription with the appropriate data.
10. Once done, the new prescription is presented to the medical professional for verification.
11. Once the user is done with verification, the user logs off (or the interface automatically times out and initiates a log off).

12. The session and connection to the database are both terminated.

--Add Illness to Patient History

1. Doctor (or other qualified medical professional) attempts to log into the system, or re-establishes an old session.
2. The interface authenticates the medical professional.
3. The central server returns an authentication token, and the interface indicates a successful login, allowing the medical professional to proceed.
4. The medical professional searches for a patient by name &/or SSN, or other data unique to one person.
5. The interface passes the query on to the server, the server returns results, and the interface displays possible matches.
6. The medical professional selects the intended patient from the list.
7. With the correct patient chosen, the interface presents a list of options. The medical professional chooses to note a new bout of an illness on the patient's record.
8. The interface presents a form to the user for the new illness, including the incident number, date of occurrence, name of prescriptions associated (if any), miscellaneous medical notes, preferred treatment, severity, and projected recovery date (if applicable), among any other proprietary information to be associated.
9. With the form filled out, the interface passes this data along to the database. The database will create a new incident entry for the given user, then fill the blank incident with the appropriate data relating to this instance of an illness.
10. Once done, the record of illness is presented to the medical professional for verification.
11. Once the user is done with verification, the user logs off (or the interface automatically times out and initiates a log off).
12. The session and connection to the database are both terminated.

--Check Illness History

1. Doctor (or other qualified medical professional) attempts to log into the system, or re-establishes an old session.
2. The interface authenticates the medical professional.
3. The central server returns an authentication token, and the interface indicates a successful login, allowing the medical professional to proceed.
4. The medical professional searches for a patient by name &/or SSN, or other data unique to one person.
5. The interface passes the query on to the server, the server returns results, and the interface displays possible matches.
6. The medical professional selects the intended patient from the list.
7. Correct patient chosen, the interface presents a list of options. The medical professional chooses to check the illness history for a given patient.

8. The interface passes this query along to the database, which then returns a number of history results, with most recent illnesses listed first. The user can request more results from the past (should more results exist), or exit from here.
9. Once the user is done viewing all the necessary history information, the user logs off (or the interface automatically times out and initiates a log off).
10. The session and connection to the database are both terminated.

--Add Other Event to History

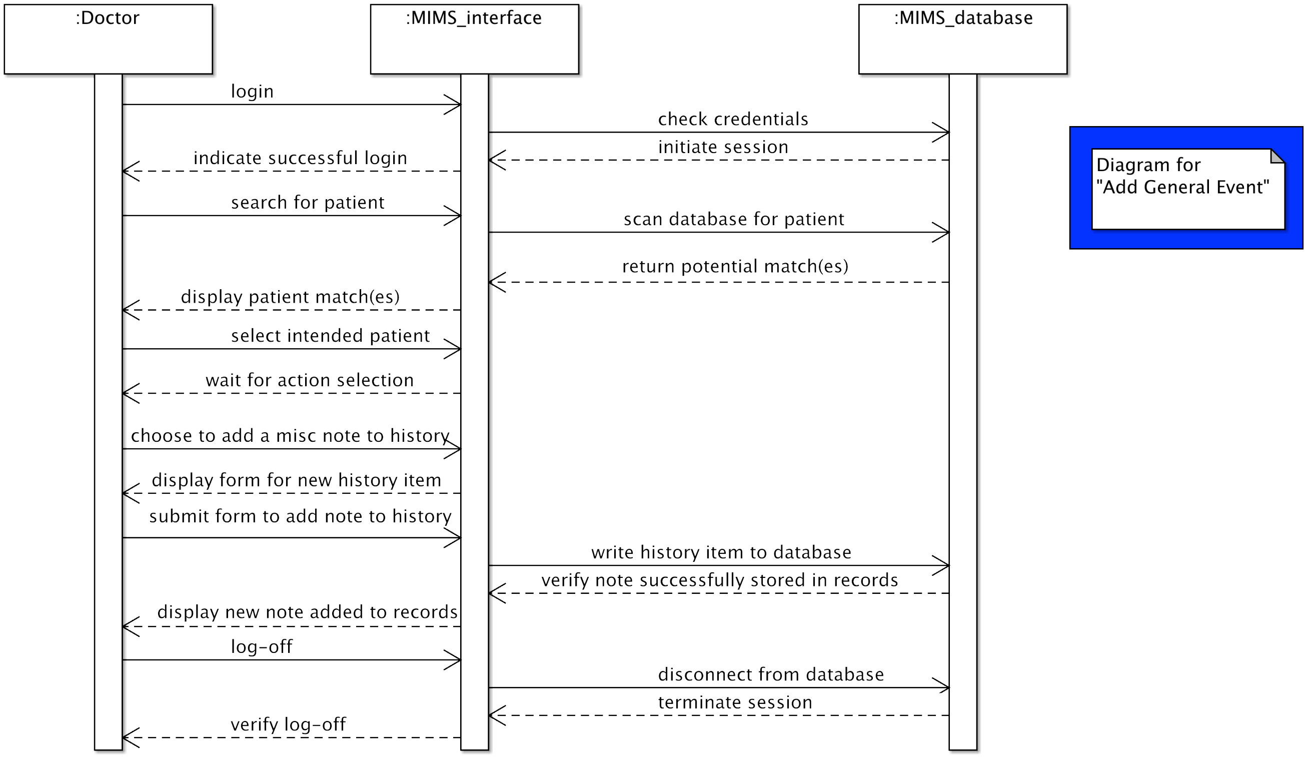
1. Doctor (or other qualified medical professional) attempts to log into the system, or re-establishes an old session.
2. The interface authenticates the medical professional.
3. The central server returns an authentication token, and the interface indicates a successful login, allowing the medical professional to proceed.
4. The medical professional searches for a patient by name &/or SSN, or other data unique to one person.
5. The interface passes the query on to the server, the server returns results, and the interface displays possible matches.
6. The medical professional selects the intended patient from the list.
7. With the correct patient chosen, the interface presents a list of options. The medical professional chooses to make a new note of an event (physical/checkup/etc) on the patient's record.
8. The interface presents a form to the user for the new event, including the event number, date of event, name of prescriptions associated (if any), miscellaneous medical notes, type of event (physical, check-up for past event, request for check-up by patient request, etc), new discoveries, changes, and other miscellaneous data that changes depending on the situation, among any other proprietary information to be associated.
9. With the form filled out, the interface passes this data along to the database. The database will create a new event entry for the given user, then fill the blank event with the appropriate data relating to this event.
10. Once done, the record of this event is presented to the medical professional for verification.
11. Once the user (doctor, etc) is done with verification, the user logs off (or the interface automatically times out and initiates a log off).
12. The session and connection to the database are both terminated.

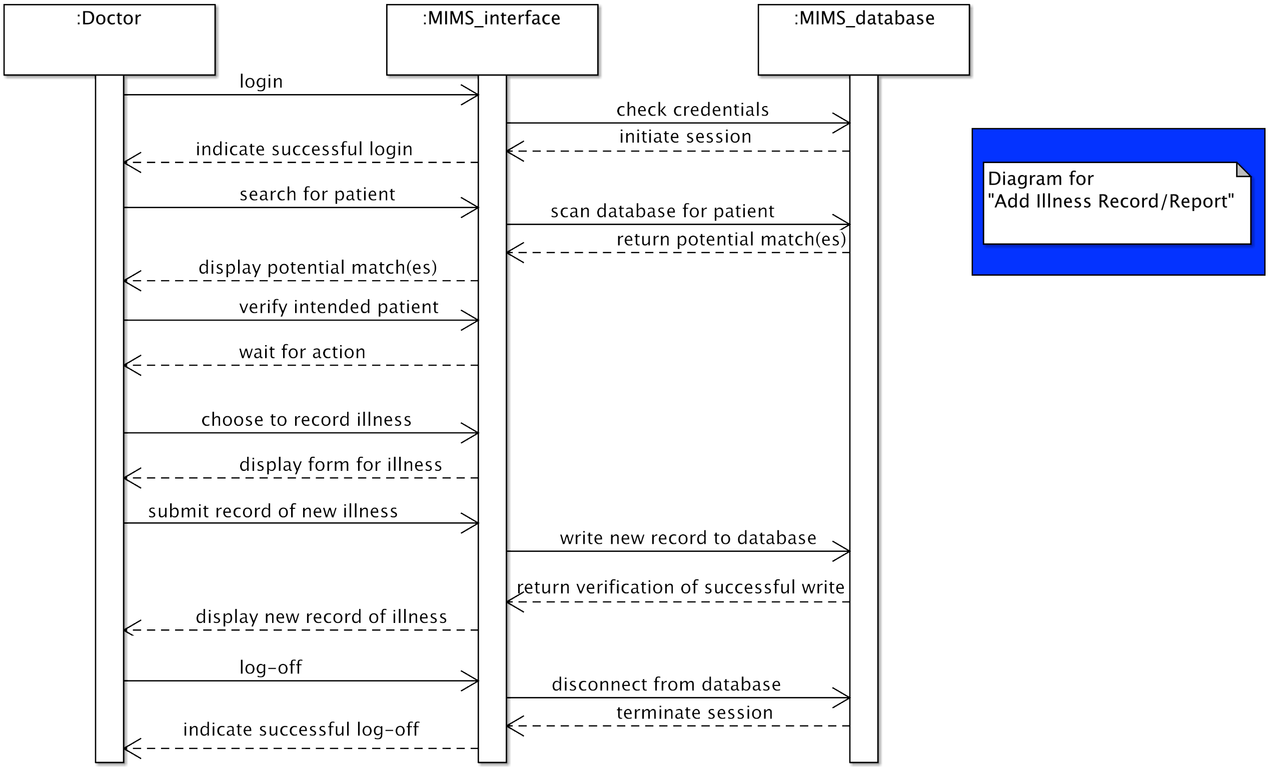
--Create New Patient (sequence diagram not pictured)

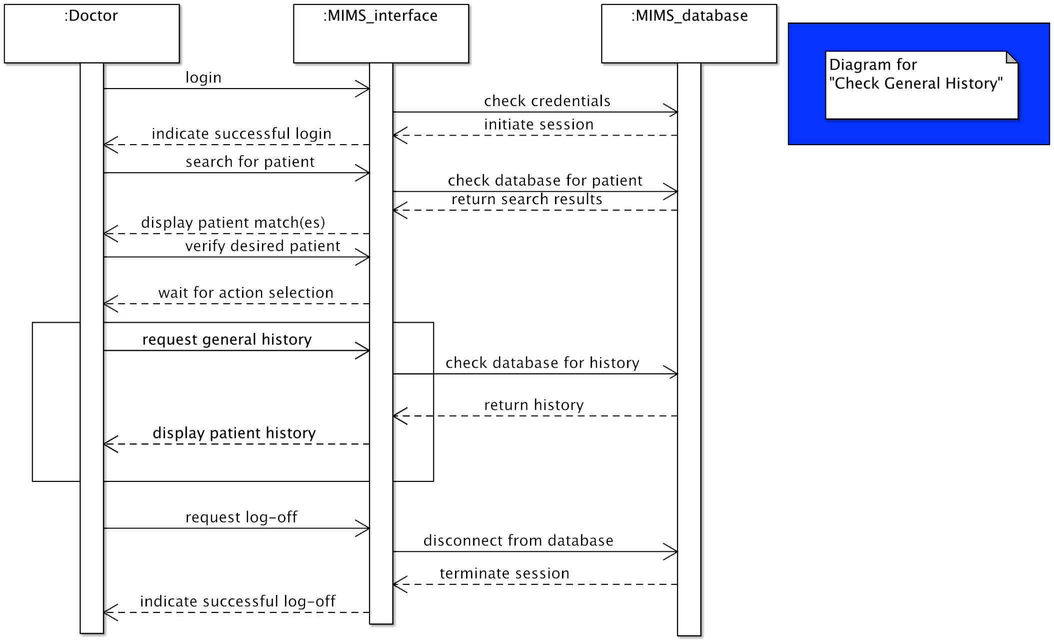
1. Doctor (or other qualified medical professional) attempts to log into the system, or re-establishes an old session.
2. The interface authenticates the medical professional.
3. The central server returns an authentication token, and the interface indicates a successful login, allowing the medical professional to proceed.
4. Rather than searching for a pre-existing patient, the doctor initiates a new patient record.

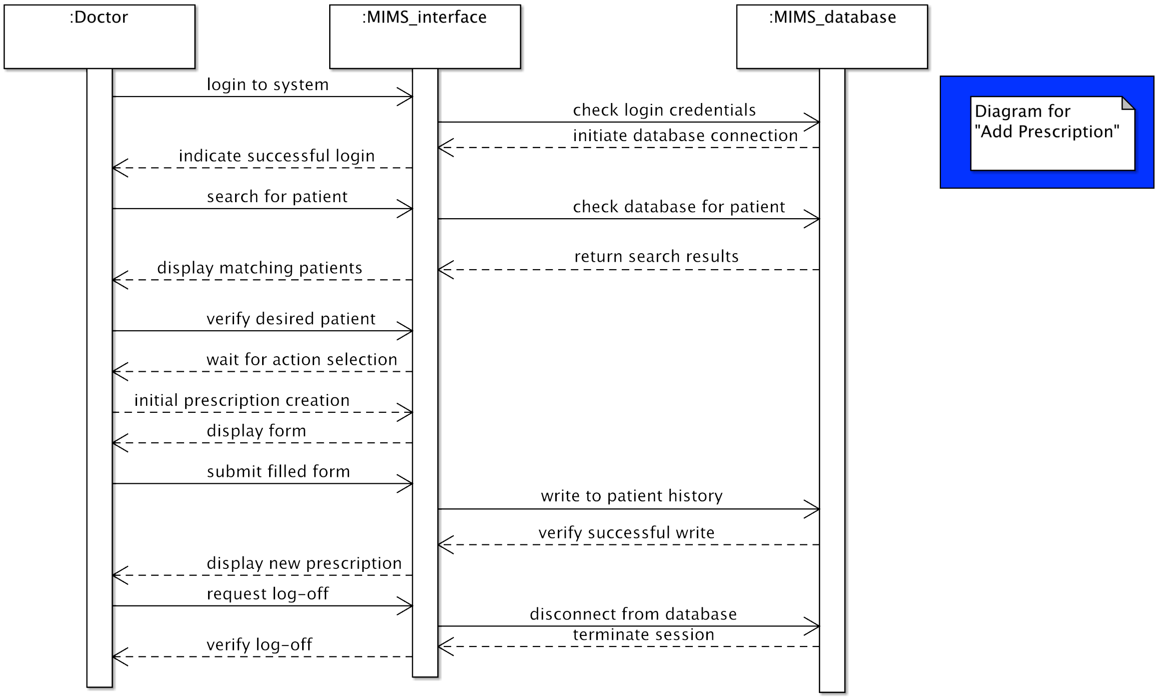
5. The interface presents a form to the doctor, first to verify that the patient indeed would be new to the system, and then to make note of any new information (name, current age, date of birth, date the new record was created in the database, SSN, associated insurance if any, and other general data).
6. The new patient record is passed to the database, and the patient is “created” in the MIMS system.
7. The interface offers to store more information manually input via keyboard or migration from another filing system.
8. The medical professional selects to add information or exit, depending.
9. The interface passes this instruction along to the database, which then verifies that the patient was successfully created, or leaves the patient record file open.
10. Depending on the situation, the interface may receive more data, then pass this data to the database for writing.
11. Once the user is done adding information (if applicable), the user logs off (or the interface automatically times out and initiates a log off).
12. The session and connection to the database are both terminated.

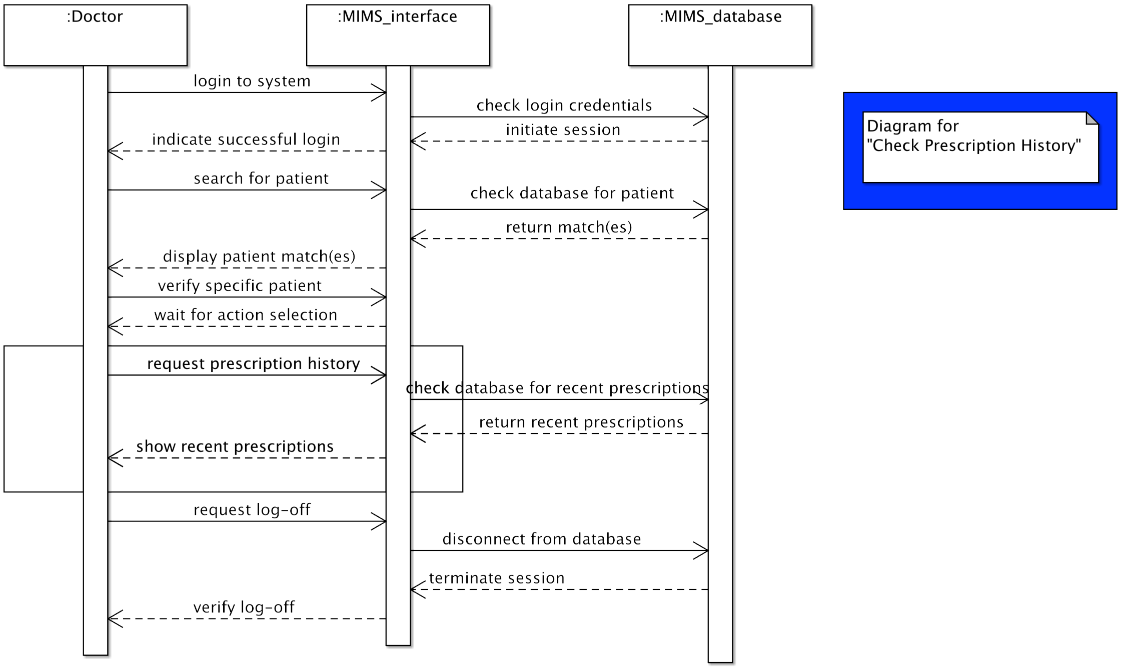
### [#4] Sample System Sequence Diagrams for first five sample sequences above


:Doctor
:MIMS_interface
:MIMS_database
Diagram for
"Add General Event"
login
check credentials
initiate session
indicate successful login
search for patient
scan database for patient
return potential match(es)
display patient match(es)
select intended patient
wait for action selection
choose to add a misc note to history
display form for new history item
submit form to add note to history
write history item to database
verify note successfully stored in records
display new note added to records
log-off
disconnect from database
terminate session
verify log-off


:Doctor
:MIMS_interface
:MIMS_database
Diagram for
"Add Illness Record/Report"
login
check credentials
initiate session
indicate successful login
search for patient
scan database for patient
return potential match(es)
display potential match(es)
verify intended patient
wait for action
choose to record illness
display form for illness
submit record of new illness
write new record to database
return verification of successful write
display new record of illness
log-off
disconnect from database
terminate session
indicate successful log-off


:Doctor
:MIMS_interface
:MIMS_database
Diagram for "Check General History"
login
check credentials
initiate session
indicate successful login
search for patient
check database for patient
return search results
display patient match(es)
verify desired patient
wait for action selection
request general history
check database for history
return history
display patient history
request log-off
disconnect from database
terminate session
indicate successful log-off


:Doctor
:MIMS_interface
:MIMS_database
Diagram for "Add Prescription"
login to system
check login credentials
initiate database connection
indicate successful login
search for patient
check database for patient
return search results
display matching patients
verify desired patient
wait for action selection
initial prescription creation
display form
submit filled form
write to patient history
verify successful write
display new prescription
request log-off
disconnect from database
terminate session
verify log-off


:Doctor
:MIMS_interface
:MIMS_database
Diagram for "Check Prescription History"
login to system
check login credentials
initiate session
indicate successful login
search for patient
check database for patient
return match(es)
display patient match(es)
verify specific patient
wait for action selection
request prescription history
check database for recent prescriptions
return recent prescriptions
show recent prescriptions
request log-off
disconnect from database
terminate session
verify log-off