CS 3354 Software Engineering
Final Project Deliverable 1

NexMed

Team: Code Byters
Alex Chan-Nui,
Cory Harris, Sophia Kobzar,
Nahome Abraham, & Jeremy Pacheco

NexMed is an app for doctors to use so patients can see their health records and for other doctors of the patient can see the records. Patients can also see their records if they need to; a patient could have forgotten when they needed to take their medicine.

- Long story short (Sophia's story)
    - My sister at an early age got a vaccine while she was taking medicine for arthritis. Because she got the vaccine, it stopped the treatment for her arthritis from working. Suppose there was better communication between my sister's doctors. My sister will not have to change to an arthritis medicine that is a lot more harmful to her body; that is why she starts with the first one in the first place. Thankfully now my sister is ok, and her arthritis is in remission.
- For a doctor to be connected to a patient
    - The doctor will send a patient a request to be added to their health team and then the patient will go into the email to accept it.
- Doctors will see anything medical that happened with the patient.
    - Vaccines, surgeries, or medication
- The doctor can only add the vaccines, surgeries, and medication etc.
    - If the patient finds an issue with their record (example: wrong medication) they should contact the doctor that was involved to change it.
- Patients can add notes if they wish.
    - Examples: Medication that did work or did not and surgery notes that patient wants to write.
- If the doctor becomes a patient to another doctor how to manage that?
    - The system will ask if the doctor need the doctor's interface or patient's
- If police, ask for records how to handle it?
    - If they have a warrant they can get the records of the patient. If they need it, they need to show it is needed. Find what we need to do to keep the records safe and what laws we must follow (HIPAA)
- Patients should be able to pay their bills on the NexMed website/app
- Patients will be able to message their doctor and vice versa on NexMed
- tasks delegated.
    - How will the interface look like for patients, doctors, and parents of minors? - Sophia Kobzar
        - Make the interface user friendly for all users.
        - Find good color schemes and models.
    - How would the setup be for doctor's accounts and patient's accounts? - Alex Chan-Nui
        - Patients can sign up like any other service. Doctors need to be check before account is live (so random people cannot have a doctors account)
    - Customer service – Nahome Abraham
        - What common issue could be possible that we could add to a help page (an FQA)
    - Patient's insurance needs to be connected to the account to help them find a doctor in their network. - Jeremy Pacheco
        - How will the billing look like?
        - How will the patient pay the doctor and how the doctor gets their money
    - Making our app able to connect with smart devices so patients and doctor can be notified when something is wrong - Cory Harris
        - Example: blood sugar levels, blood pressure levels etc.
        - When do people(patients and doctors) need to be notified when something isn't right?
        - How will patients add device to account & app?

Instructor feedback about NexMed: “A lovely topic!! Once complete, it will save a lot of time and effort for doctors and patients who are to meet for a sustained healthcare system. Please consider implementing it fully if you can. No pressure w.r.t. grade on implementation. In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how.
Fair delegation of tasks.
Please share this feedback with your group members.
You are good to go. Have fun with the project and hope everyone enjoys the collaboration.”

Git repository URL: https://github.com/sophiakobzar/3354-codeByters.git

Delegation of tasks of project deliverable 1:

- Alex Chan-Nui
  - Question: six
- Cory Harris
  - Question: eight
  - Commit file to GIT repository
- Jeremy Pacheco
  - Question: nine
  - Commit file to GIT repository
- Nahome Abraham
  - Question: seven
- Sophia Kobzar
  - Questions: 1 & 3-5
  - Made the Git repository and committed to Git repository

Software process model that is employed in the project and why:

- The Waterfall model is what the group choose:
  - In the waterfall model everyone gets up to speed quickly, schedules are kept, there are no financial surprises, testing is done easily, and the outcome is clear; so, you get what you planned. The waterfall model can also be challenging to define needs, lacks flexibility, and longer delivery time.. Often the most criticized aspect of waterfall development; the customer does not know what they want. It is important that software developers can guide and advise clients effectively to avoid problems later.

Software Requirements:

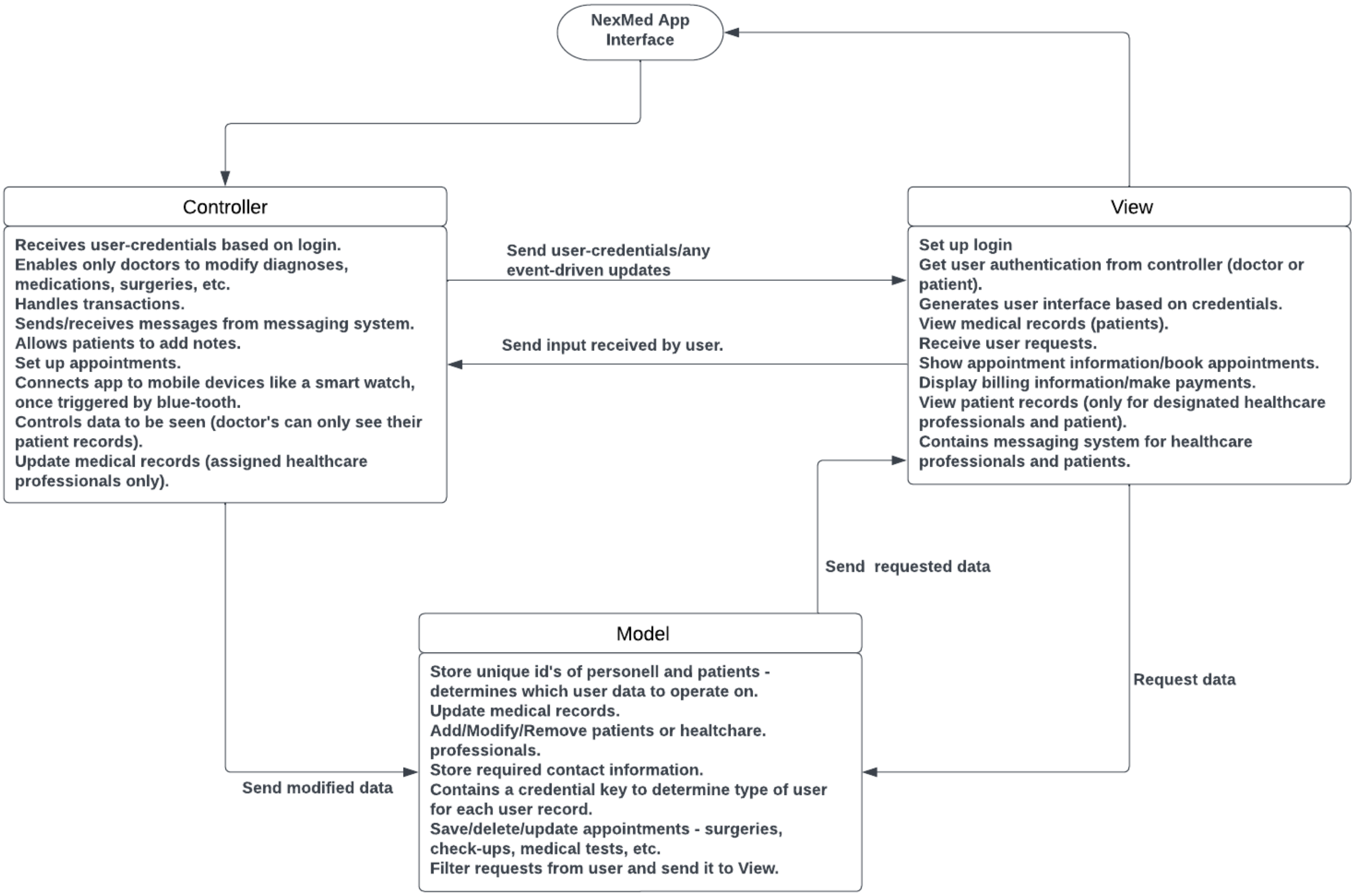
- Functional requirements
  - Doctors should be able to connect to patients’ profile.
  - Patients and their doctors should see the patient’s profile. Doctors should be able to see their patients only.
  - Users(doctors, patients, patient’s guardian if patient minor) should be able to sign into their account. If a doctor becomes a patient the use interface will be different.
  - Minors can see their profile and their guardian can to until they separate. Guardians and minor patients can separate when the minor patient turns 18 years old, or minor is emancipated.
  - Patient’s profile (medications, surgeries, allergies, etc.) should be displayed on their account in a user-friendly manner.

  - The information about the patient’s medication that is needed:
    - Information of the patient (Name, DOB, home address, contact information)
    - If the medication is currently taken.
    - Size of pills (if applicable could be liquid)
    - size of bottle (if applicable could be pills)
    - Dose size (number of pills or liquid dose)
    - When should the patient take the medicine?
    - Anything notes the patient needs to know (side effects, only take medicine after/before eating, etc.)
    - Which pharmacy was the medicine sent to?
  - The information about the patient’s surgery that is needed:
    - When was the surgery (time and date)
    - Where was the surgery (which hospital)
    - What is the surgery the patient had?
    - What did the surgeon do?
    - Contact information of the surgeon team

- Non-functional requirements
  - Product requirement
    - Efficiency requirements
      - Performance requirements
        - For read requests max 100 milliseconds latency
        - For write requests(create or update data) we want to keep under 300 milliseconds latency.
      - Space requirements
        - There should be enough space in the server for patient data. Also, copy need to be made of the data.
        - We will utilize AWS infrastructure to run frontend, backend and database services.
    - Dependability requirements
      - No matter the time or day the patient’s data should be reachable.
    - Security requirements
      - Only appropriate staff members should be able to see the patient’s data.
    - Usability requirements
      - Any laptop or Mac can access the website.
      - IOS and android can download the app.

  - Organizational requirements
    - Environmental requirements
      - The servers should be kept in a clean cool room.
    - Operational requirements
      - Someone needs to be hired to maintain the server room.
    - Development requirements
      - Developers need to be hired to do regular updates.

- External requirements
    - Regulatory requirements
        - Random people should not be able to access the server room.
    - Ethical requirements
        - Doctors should not be able to share the patient’s data with unapproved sources.
    - Legislative requirements
        - Accounting requirements
            - Keep track of the current clients and getting new ones.
        - safety/security requirements
            - Only appropriate members can access the data room.
            - Apps also must follow HIPPA laws.



Controller

Receives user-credentials based on login.
Enables only doctors to modify diagnoses, medications, surgeries, etc.
Handles transactions.
Sends/receives messages from messaging system.
Allows patients to add notes.
Set up appointments.
Connects app to mobile devices like a smart watch, once triggered by blue-tooth.
Controls data to be seen (doctor's can only see their patient records).
Update medical records (assigned healthcare professionals only).

View

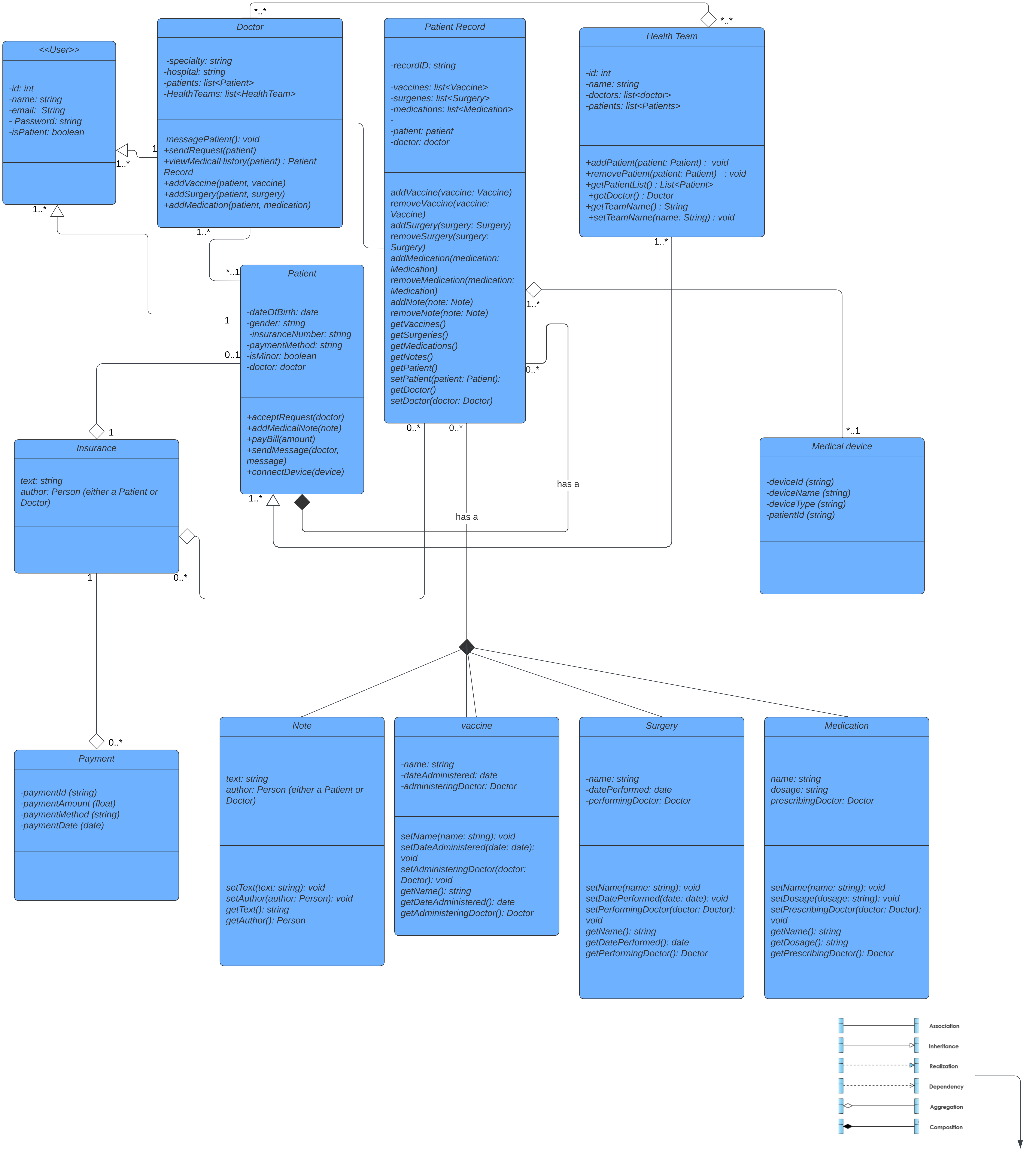
Set up login
Get user authentication from controller (doctor or patient).
Generates user interface based on credentials.
View medical records (patients).
Receive user requests.
Show appointment information/book appointments.
Display billing information/make payments.
View patient records (only for designated healthcare professionals and patient).
Contains messaging system for healthcare professionals and patients.

Model

Store unique id's of personell and patients - determines which user data to operate on.
Update medical records.
Add/Modify/Remove patients or healtchare. professionals.
Store required contact information.
Contains a credential key to determine type of user for each user record.
Save/delete/update appointments - surgeries, check-ups, medical tests, etc.
Filter requests from user and send it to View.

ClassDiagram_NexMed

cory harris | March 24, 2023



<<User>>
-id: int
-name: string
-email: String
- Password: string
-isPatient: boolean

Doctor
-specialty: string
-hospital: string
-patients: list<Patient>
-HealthTeams: list<HealthTeam>

messagePatient(): void
+sendRequest(patient)
+viewMedicalHistory(patient) : Patient Record
+addVaccine(patient, vaccine)
+addSurgery(patient, surgery)
+addMedication(patient, medication)

Patient Record
-recordID: string

-vaccines: list<Vaccine>
-surgeries: list<Surgery>
-medications: list<Medication>
-
-patient: patient
-doctor: doctor

addVaccine(vaccine: Vaccine)
removeVaccine(vaccine: Vaccine)
addSurgery(surgery: Surgery)
removeSurgery(surgery: Surgery)
addMedication(medication: Medication)
removeMedication(medication: Medication)
addNote(note: Note)
removeNote(note: Note)
getVaccines()
getSurgeries()
getMedications()
getNotes()
getPatient()
setPatient(patient: Patient):
getDoctor()
setDoctor(doctor: Doctor)

Health Team
-id: int
-name: string
-doctors: list<doctor>
-patients: list<Patients>

+addPatient(patient: Patient) : void
+removePatient(patient: Patient) : void
+getPatientList() : List<Patient>
+getDoctor() : Doctor
+getTeamName() : String
+setTeamName(name: String) : void

Patient
-dateOfBirth: date
-gender: string
-insuranceNumber: string
-paymentMethod: string
-isMinor: boolean
-doctor: doctor

+acceptRequest(doctor)
+addMedicalNote(note)
+payBill(amount)
+sendMessage(doctor, message)
+connectDevice(device)

Insurance
text: string
author: Person (either a Patient or Doctor)

Medical device
-deviceId (string)
-deviceName (string)
-deviceType (string)
-patientId (string)

Payment
-paymentId (string)
-paymentAmount (float)
-paymentMethod (string)
-paymentDate (date)

Note
text: string
author: Person (either a Patient or Doctor)

setText(text: string): void
setAuthor(author: Person): void
getText(): string
getAuthor(): Person

vaccine
-name: string
-dateAdministered: date
-administeringDoctor: Doctor

setName(name: string): void
setDateAdministered(date: date): void
setAdministeringDoctor(doctor: Doctor): void
getName(): string
getDateAdministered(): date
getAdministeringDoctor(): Doctor

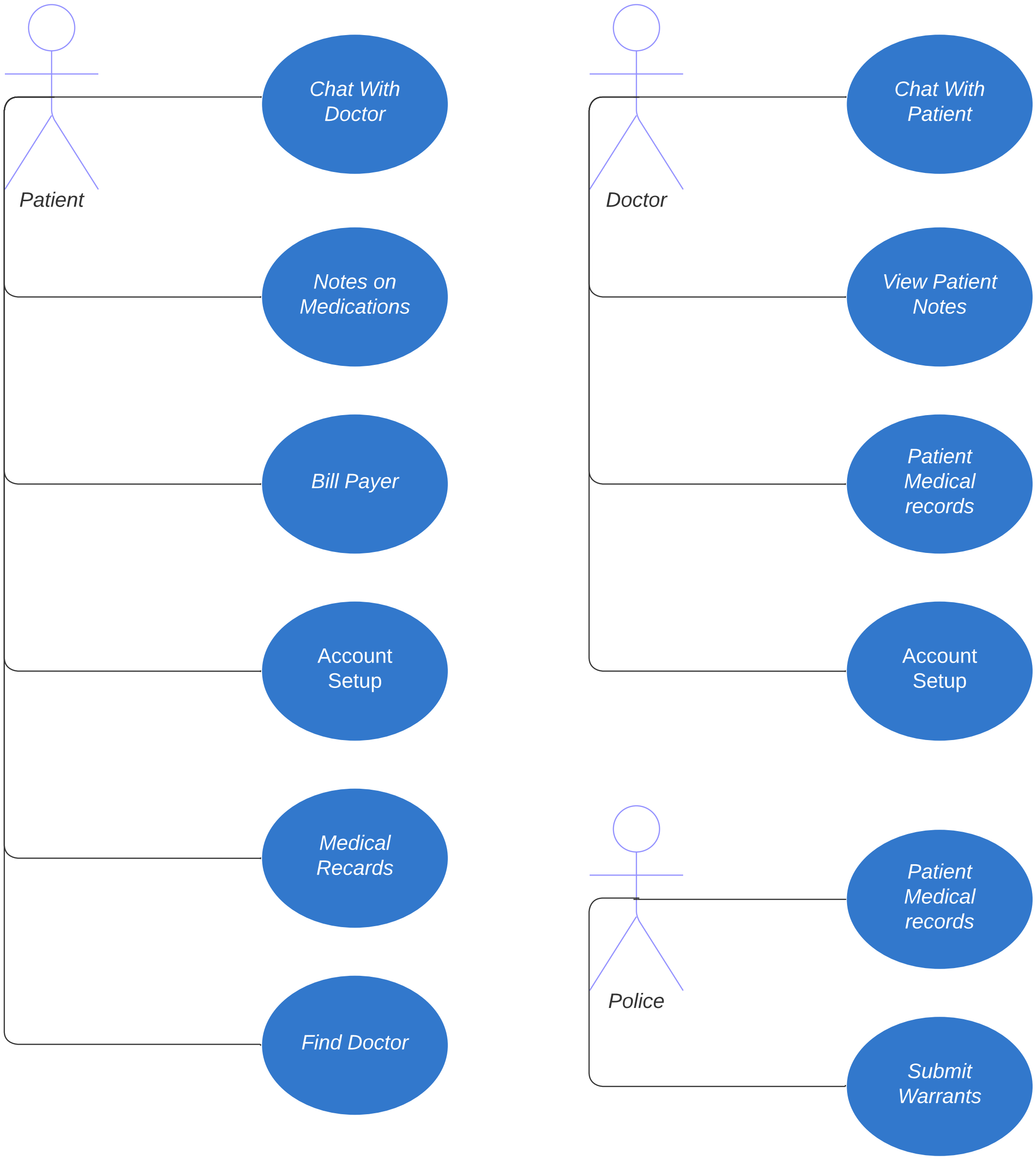
Surgery
-name: string
-datePerformed: date
-performingDoctor: Doctor

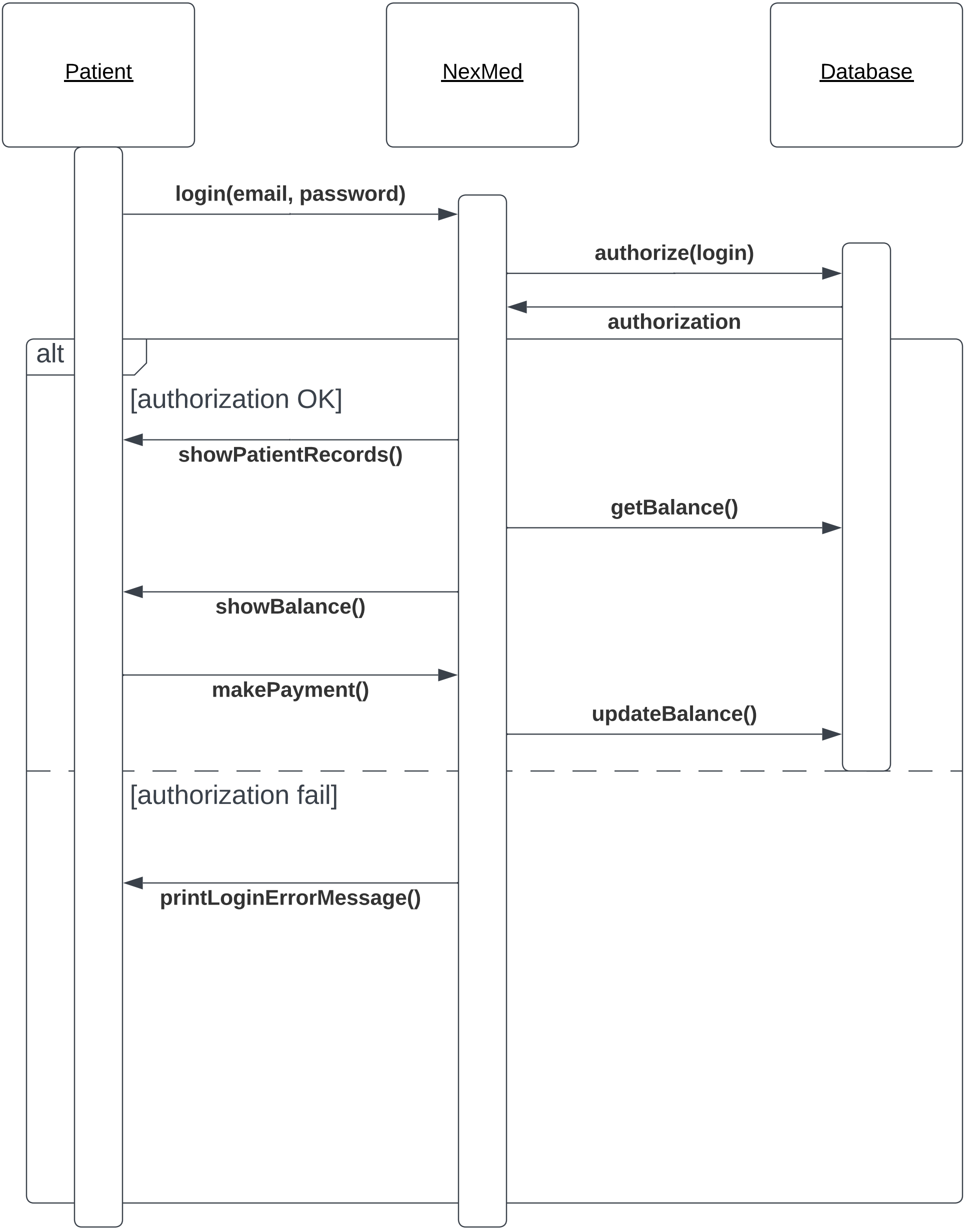
setName(name: string): void
setDatePerformed(date: date): void
setPerformingDoctor(doctor: Doctor): void
getName(): string
getDatePerformed(): date
getPerformingDoctor(): Doctor

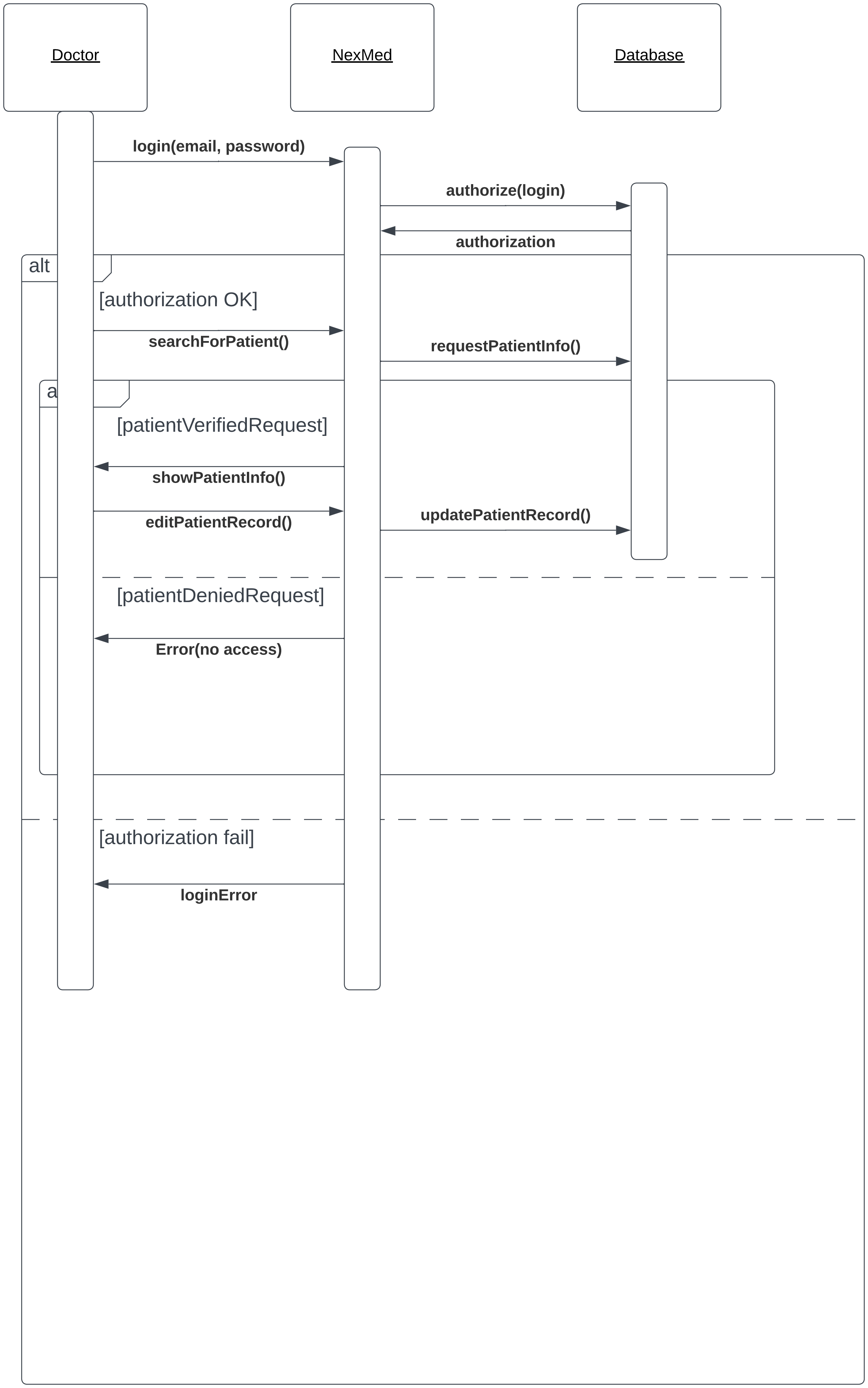
Medication
name: string
dosage: string
prescribingDoctor: Doctor

setName(name: string): void
setDosage(dosage: string): void
setPrescribingDoctor(doctor: Doctor): void
getName(): string
getDosage(): string
getPrescribingDoctor(): Doctor

SOURCE:https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/


Patient
Chat With Doctor
Notes on Medications
Bill Payer
Account Setup
Medical Recards
Find Doctor
Doctor
Chat With Patient
View Patient Notes
Patient Medical records
Account Setup
Police
Patient Medical records
Submit Warrants


Patient
NexMed
Database
login(email, password)
authorize(login)
authorization
alt
[authorization OK]
showPatientRecords()
getBalance()
showBalance()
makePayment()
updateBalance()
[authorization fail]
printLoginErrorMessage()


Doctor
NexMed
Database
login(email, password)
authorize(login)
authorization
alt
[authorization OK]
searchForPatient()
requestPatientInfo()
a
[patientVerifiedRequest]
showPatientInfo()
editPatientRecord()
updatePatientRecord()
[patientDeniedRequest]
Error(no access)
[authorization fail]
loginError