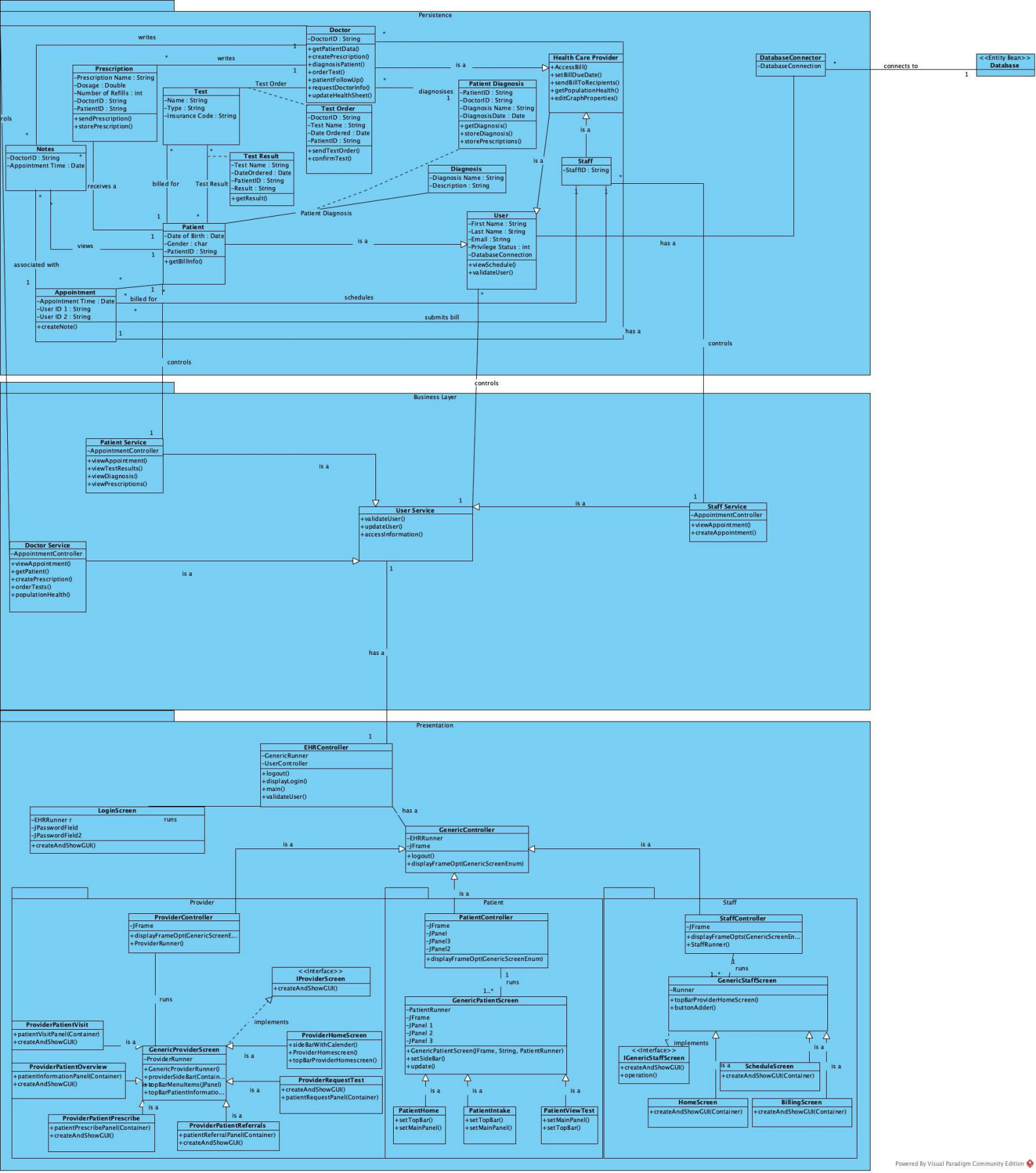
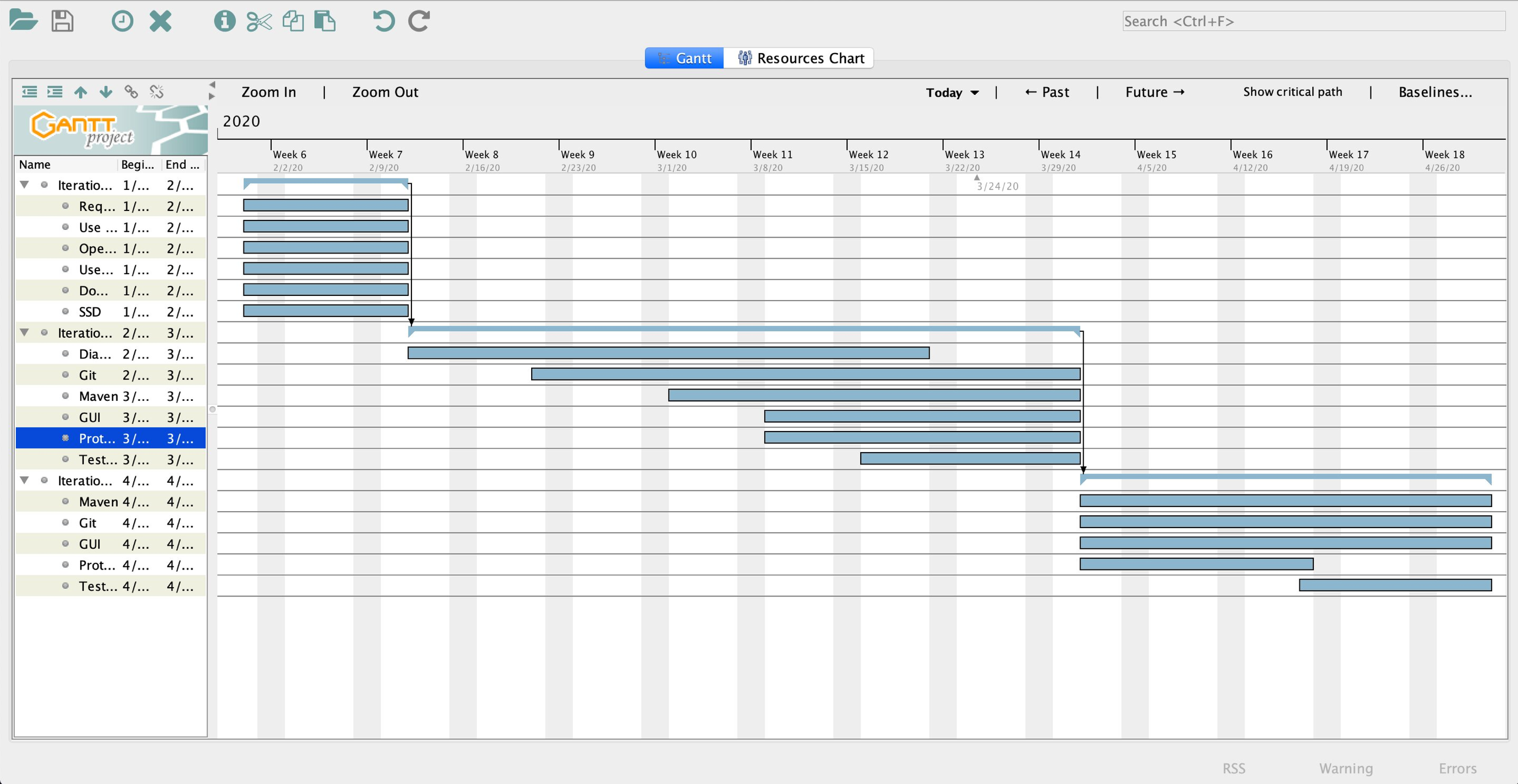

Persistence
Doctor
-DoctorID : String
+getPatientData()
+createPrescription()
+diagnosisPatient()
+orderTest()
+patientFollowUp()
+requestDoctorInfo()
+updateHealthSheet()
writes
Prescription
-Prescription Name : String
-Dosage : Double
-Number of Refills : int
-DoctorID : String
-PatientID : String
+sendPrescription()
+storePrescription()
Test
-Name : String
-Type : String
-Insurance Code : String
Test Order
-DoctorID : String
-Test Name : String
-Date Ordered : Date
-PatientID : String
+sendTestOrder()
+confirmTest()
Health Care Provider
+AccessBill()
+setBillDueDate()
+sendBillToRecipients()
+getPopulationHealth()
+editGraphProperties()
is a
Patient Diagnosis
-PatientID : String
-DoctorID : String
-Diagnosis Name : String
-DiagnosisDate : Date
+getDiagnosis()
+storeDiagnosis()
+storePrescriptions()
diagnosises
DatabaseConnector
-DatabaseConnection
connects to
<<Entity Bean>>
Database
Notes
-DoctorID : String
-Appointment Time : Date
receives a
billed for
Test Result
-Test Name : String
-DateOrdered : Date
-PatientID : String
-Result : String
+getResult()
Diagnosis
-Diagnosis Name : String
-Description : String
Staff
-StaffID : String
Patient
-Date of Birth : Date
-Gender : char
-PatientID : String
+getBillInfo()
views
associated with
User
-First Name : String
-Last Name : String
-Email : String
-Privilege Status : int
-DatabaseConnection
+viewSchedule()
+validateUser()
has a
Appointment
-Appointment Time : Date
-User ID 1 : String
-User ID 2 : String
+createNote()
schedules
submits bill
controls
Business Layer
Patient Service
-AppointmentController
+viewAppointment()
+viewTestResults()
+viewDiagnosis()
+viewPrescriptions()
User Service
+validateUser()
+updateUser()
+accessInformation()
Staff Service
-AppointmentController
+viewAppointment()
+createAppointment()
Doctor Service
-AppointmentController
+viewAppointment()
+getPatient()
+createPrescription()
+orderTests()
+populationHealth()
Presentation
EHRController
-GenericRunner
-UserController
+logout()
+displayLogin()
+main()
+validateUser()
LoginScreen
-EHRRunner r
-JPasswordField
-JPasswordField2
+createAndShowGUI()
runs
GenericController
-EHRRunner
-JFrame
+logout()
+displayFrameOpt(GenericScreenEnum)
Provider
ProviderController
-JFrame
+displayFrameOpt(GenericScreenE...
+ProviderRunner()
<<Interface>>
IProviderScreen
+createAndShowGUI()
implements
ProviderPatientVisit
+patientVisitPanel(Container)
+createAndShowGUI()
GenericProviderScreen
-ProviderRunner
+GenericProviderRunner()
+providerSideBar(Contain...
+topBarMenuItems(JPanel)
+topBarPatientInformatio...
ProviderHomeScreen
+sideBarWithCalender()
+ProviderHomescreen()
+topBarProviderHomescreen()
ProviderPatientOverview
+patientInformationPanel(Container)
+createAndShowGUI()
ProviderRequestTest
+createAndShowGUI()
+patientRequestPanel(Container)
ProviderPatientPrescribe
+patientPrescribePanel(Container)
+createAndShowGUI()
ProviderPatientReferrals
+patientReferralPanel(Container)
+createAndShowGUI()
Patient
PatientController
-JFrame
-JPanel
-JPanel3
-JPanel2
+displayFrameOpt(GenericScreenEnum)
GenericPatientScreen
-PatientRunner
-JFrame
-JPanel 1
-JPanel 2
-JPanel 3
+GenericPatientScreen()Frame, String, PatientRunner)
+setSideBar()
+update()
PatientHome
+setTopBar()
+setMainPanel()
PatientIntake
+setTopBar()
+setMainPanel()
PatientViewTest
+setMainPanel()
+setTopBar()
Staff
StaffController
-JFrame
+displayFrameOpts(GenericScreenEn...
+StaffRunner()
GenericStaffScreen
-Runner
+topBarProviderHomeScreen()
+buttonAdder()
<<Interface>>
IGenericStaffScreen
+createAndShowGUI()
+operation()
ScheduleScreen
+createAndShowGUI(Container)
HomeScreen
+createAndShowGUI(Container)
BillingScreen
+createAndShowGUI(Container)
is a
has a
controls
Powered By Visual Paradigm Community Edition


Search <Ctrl+F>
Gantt
Resources Chart
Zoom In | Zoom Out
Today | ← Past | Future →
Show critical path | Baselines...
GANTT project
2020
Week 6 2/2/20
Week 7 2/9/20
Week 8 2/16/20
Week 9 2/23/20
Week 10 3/1/20
Week 11 3/8/20
Week 12 3/15/20
Week 13 3/22/20
3/24/20
Week 14 3/29/20
Week 15 4/5/20
Week 16 4/12/20
Week 17 4/19/20
Week 18 4/26/20
Name Begi... End ...
Iteratio... 1/... 2/...
Req... 1/... 2/...
Use ... 1/... 2/...
Ope... 1/... 2/...
Use... 1/... 2/...
Do... 1/... 2/...
SSD 1/... 2/...
Iteratio... 2/... 3/...
Dia... 2/... 3/...
Git 2/... 3/...
Maven 3/... 3/...
GUI 3/... 3/...
Prot... 3/... 3/...
Test... 3/... 3/...
Iteratio... 4/... 4/...
Maven 4/... 4/...
Git 4/... 4/...
GUI 4/... 4/...
Prot... 4/... 4/...
Test... 4/... 4/...
RSS
Warning
Errors

GRASP

1. getAppointmentDate:
    - Location: Appointment
    - Principle/Pattern: Information Expert
        - Responsible for holding information for other classes; holds the time of the appointment for the notes and the billing information for the patient.

2. getDescription / getName:
    - Location: Diagnosis
    - Principle/Pattern: Information Expert
        - Responsible for holding information about diagnoses, these methods relay the information the class knows about the ailment of the patient, its name and description.

3. createPrescription / orderTest:
    - Location: Doctor
    - Principle/Pattern: Creator
        - Responsible for using the input of the doctor in order to create test and prescription objects

4. updateHealthSheet
    - Location: Doctor
    - Principle/Pattern: Controller
        - Responsible for using the information passed to the Doctor class to edit the patient's health sheet, which is outside of the doctor class.

5. sendBillToRecipients:
    - Location: Health Care Provider
    - Principle/Pattern: Controller
        - This operation is responsible for gathering information from the appointment and providing it to the necessary parties.

6. editGraphProperties:
    - Location: Health Care Provider
    - Principle/Pattern: Information Expert
        - This operation is responsible for changing the graph to represent the way it is to be displayed, which it knows.

7. getBillInfo:
    - Location: Patient
    - Principle/Pattern: Information Expert
        - Responsible for knowing the information regarding the bill the patient must pay, and potentially making that information available to the patient.

8. getPrescription:
    - Location: Prescription
    - Principle/Pattern: Information Expert
        - Responsible for knowing the information regarding the medication, number of refills, patient ID, and doctor ID along with the dosage. This information can be accessed by the doctor and patient.

9. storeDiagnosis / storePrescriptions:
    - Location: Patient Diagnosis
    - Principle/Pattern: Controller
        - Responsible for assigning an existing Diagnosis or Prescription to a Patient based on the input of the Doctor.

10. createAppointment / sendTestOrder:
    - Location: Staff
    - Principle/Pattern: Creator
        - Responsible for creating Appointments and Tests

11. getTestName / getInsuranceCode:
    - Location: Test
    - Principle/Pattern: Information Expert
        - Responsible for storing data necessary to be able to perform a test on a Patient in a lab.

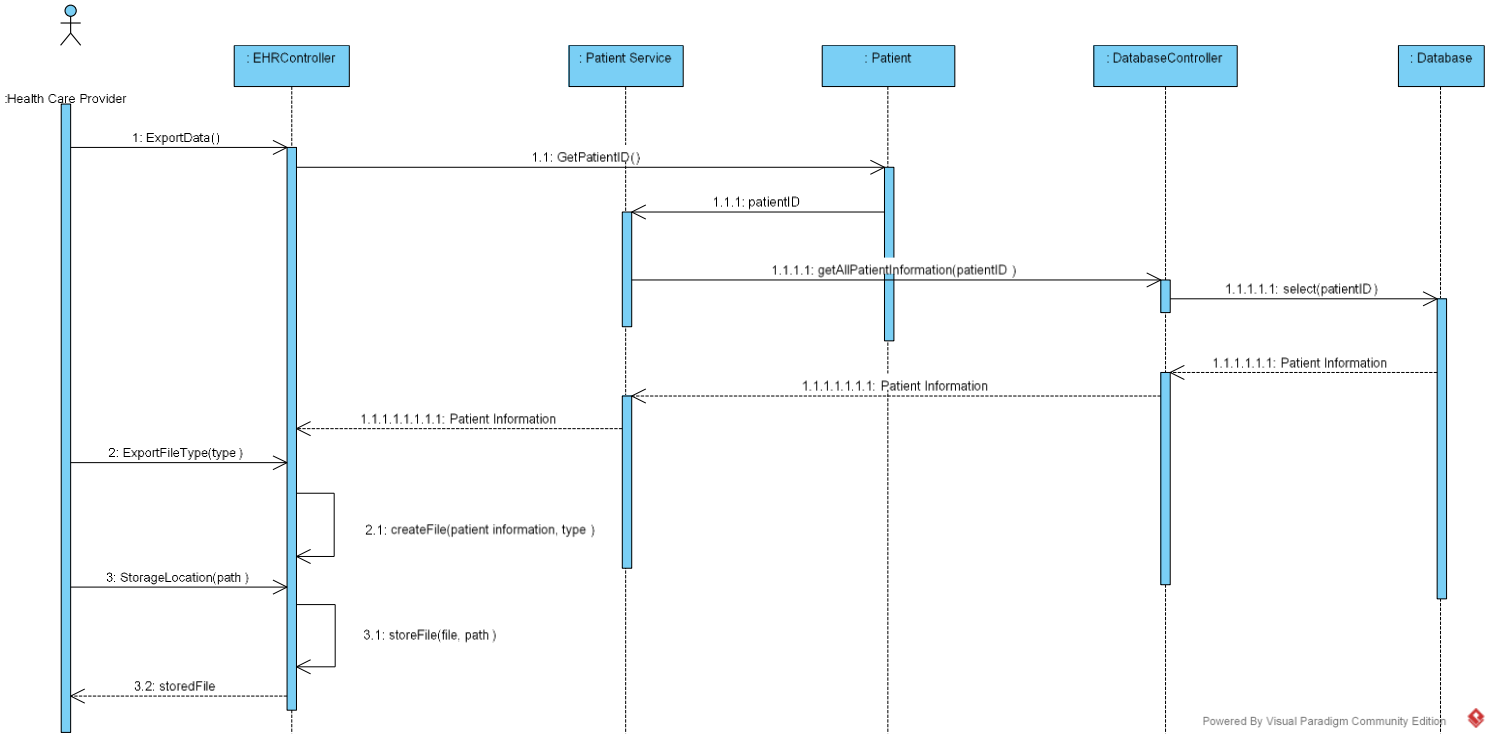
12. sendTestOrder / confirmTest:
    - Location: Test Order
    - Principle/Pattern: Controller
        - Responsible for assigning an existing Test to a Patient based on input from a Staff member.

13. getTestResult:
    - Location: Test Result
    - Principle/Pattern: Controller
        - Responsible for taking input from the Test which was performed and relaying the information to the Patient.

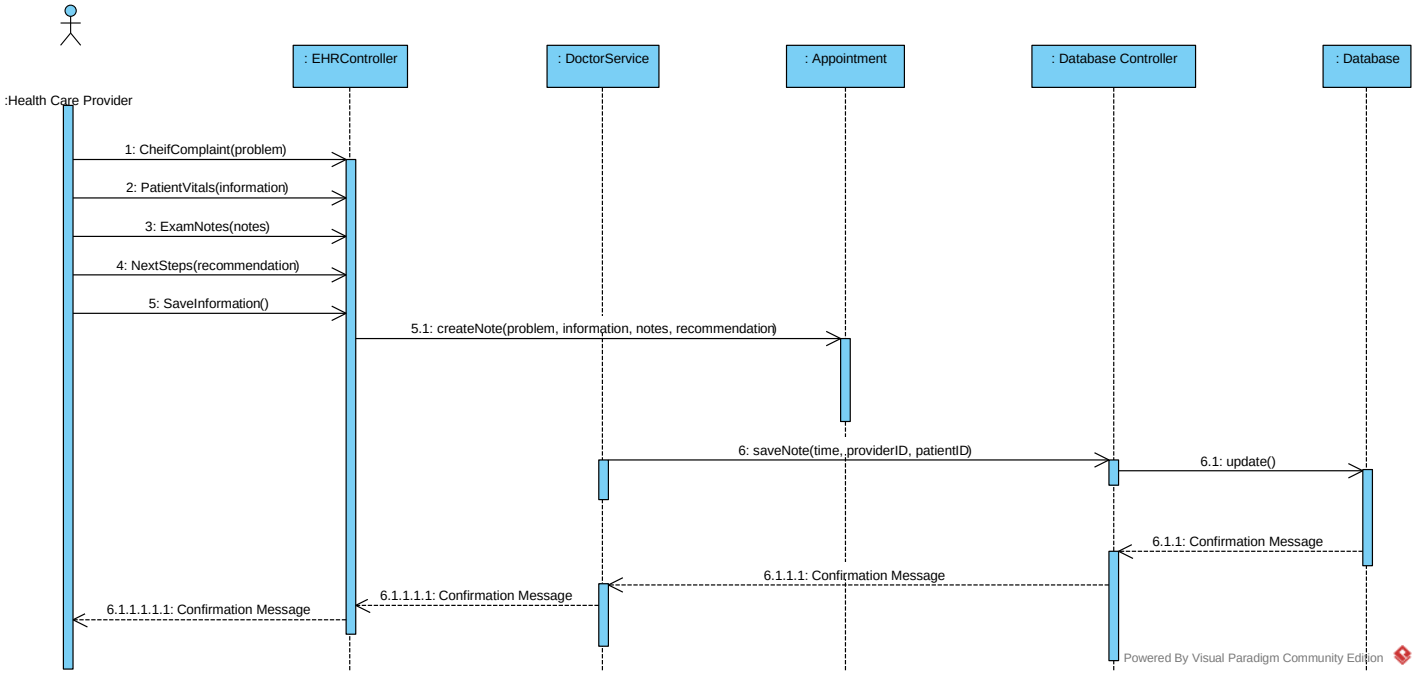
14. viewSchedule:
    - Location: User
    - Principle/Pattern: Low Coupling, High Cohesion, Polymorphism, Information Expert
        - Useful for polymorphism, allows different types of users to view their own personalized schedule, being either Doctor, Patient, or Staff

15. SQLConnection():
    - Location: SQLConnection
    - Principle/Pattern: Low Coupling, Controller
        - Receives information from a database and initializes data in other classes

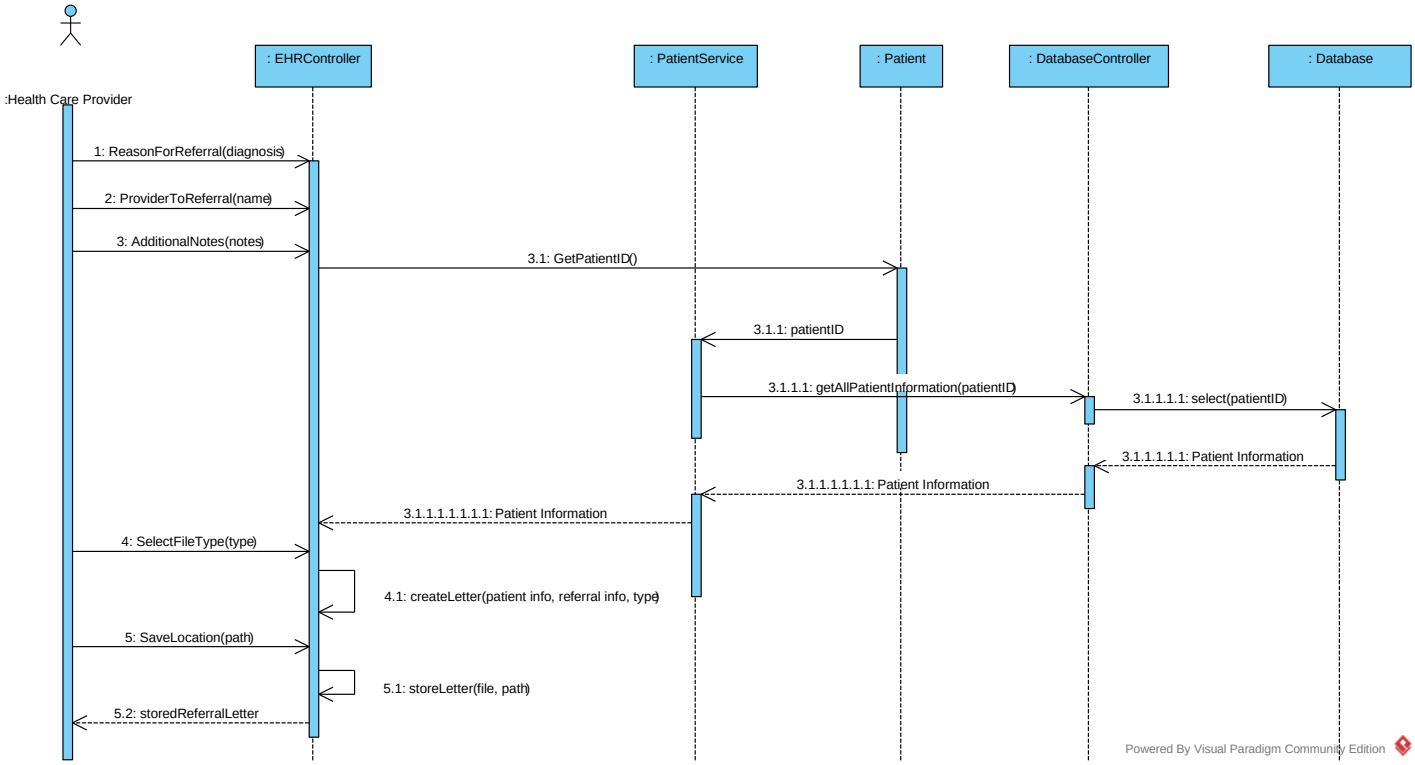
Sequence Diagram: Export Data



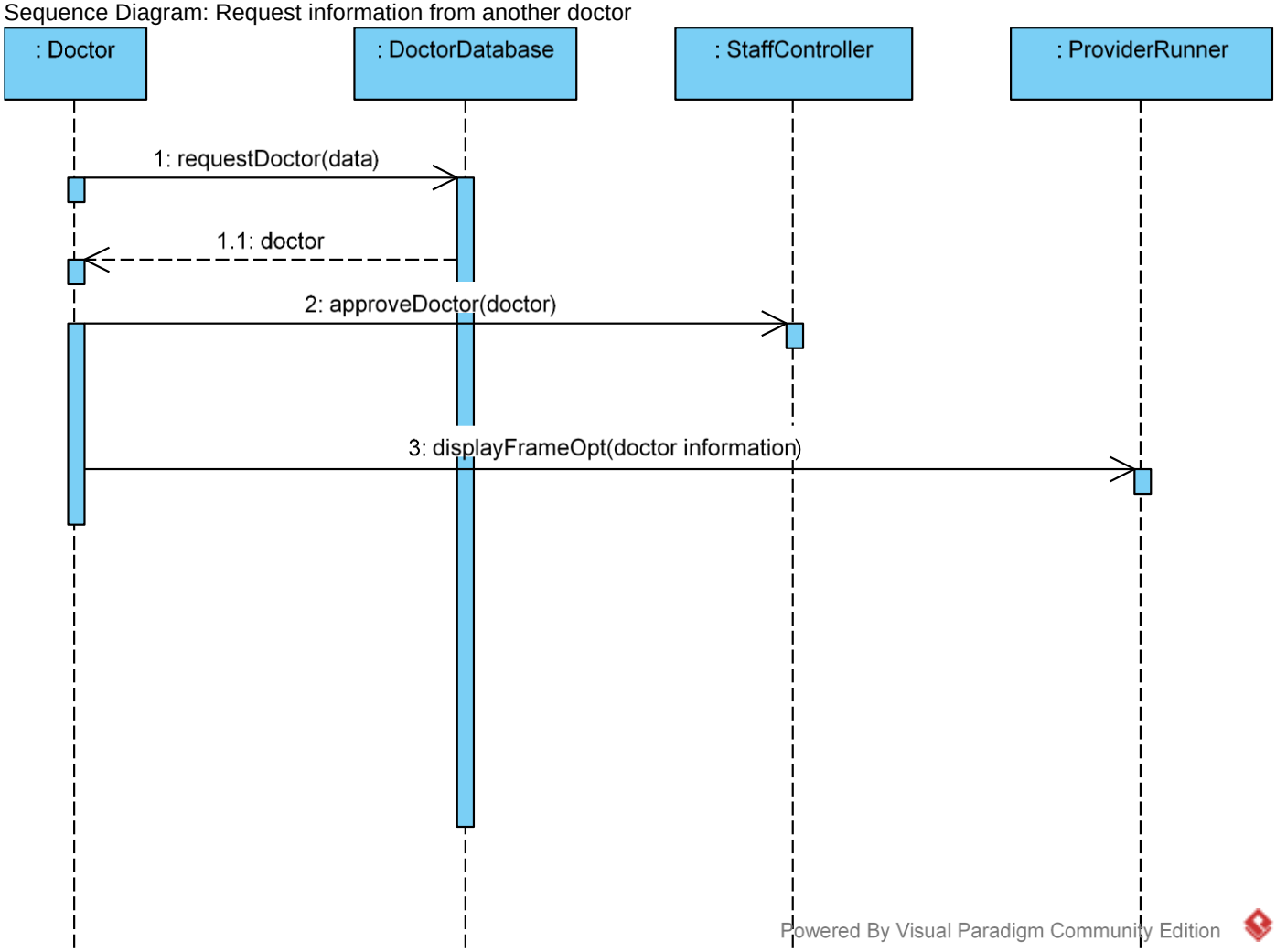
Sequence Diagram: Provider Notes



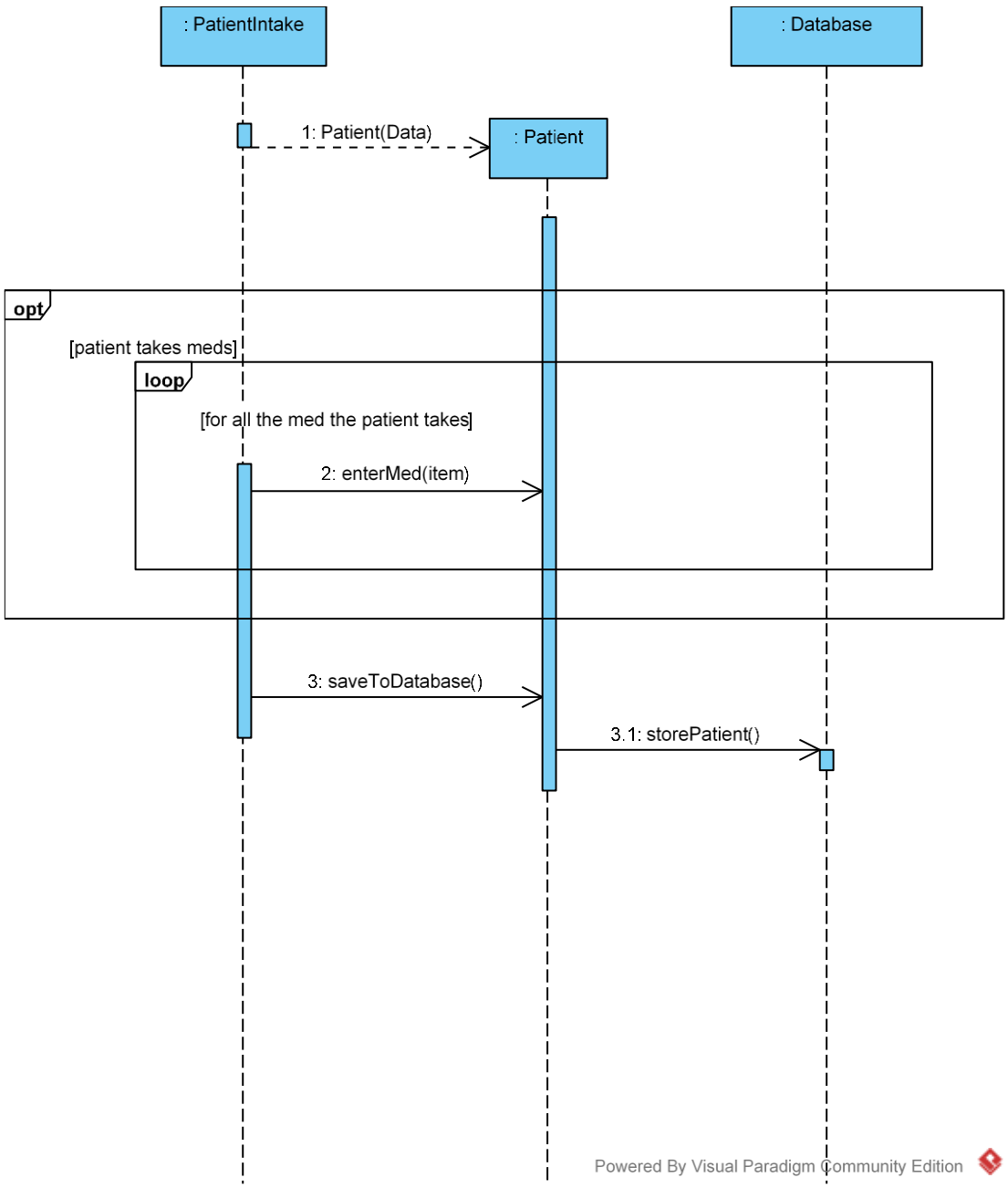
Sequence Diagram: Referral Letter



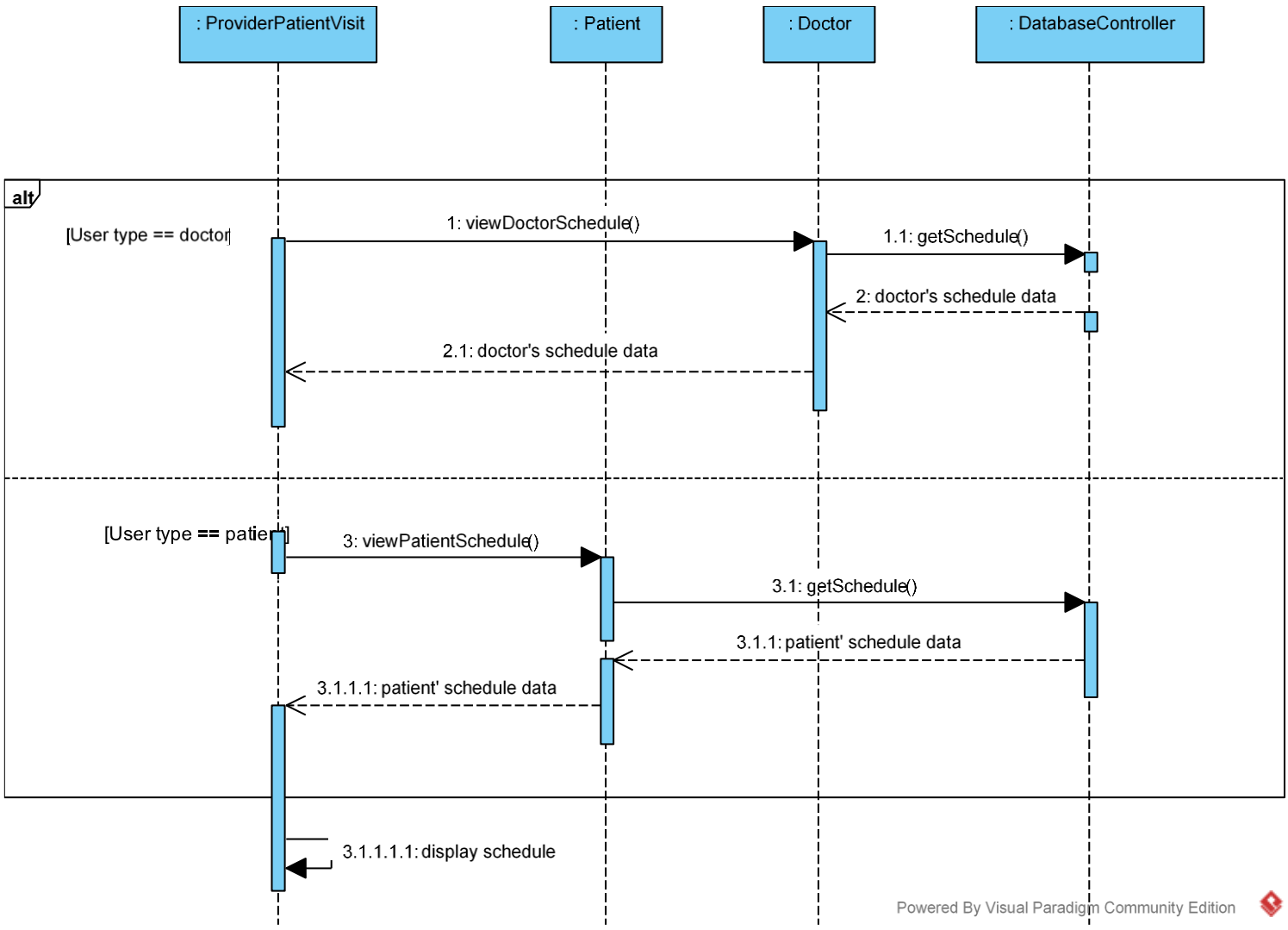
Sequence Diagram: Request information from another doctor



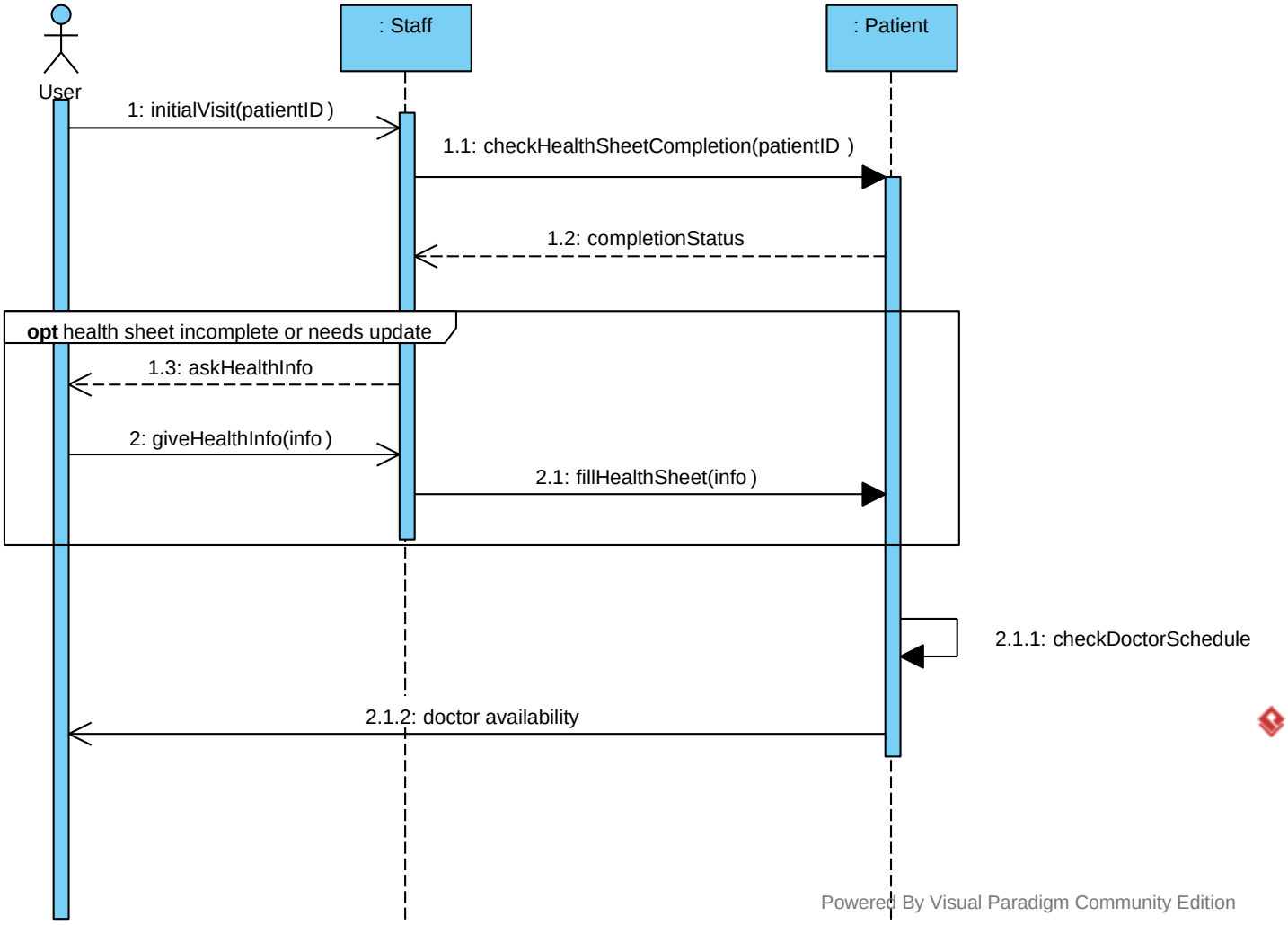
Sequence Diagram: Signing up for patient



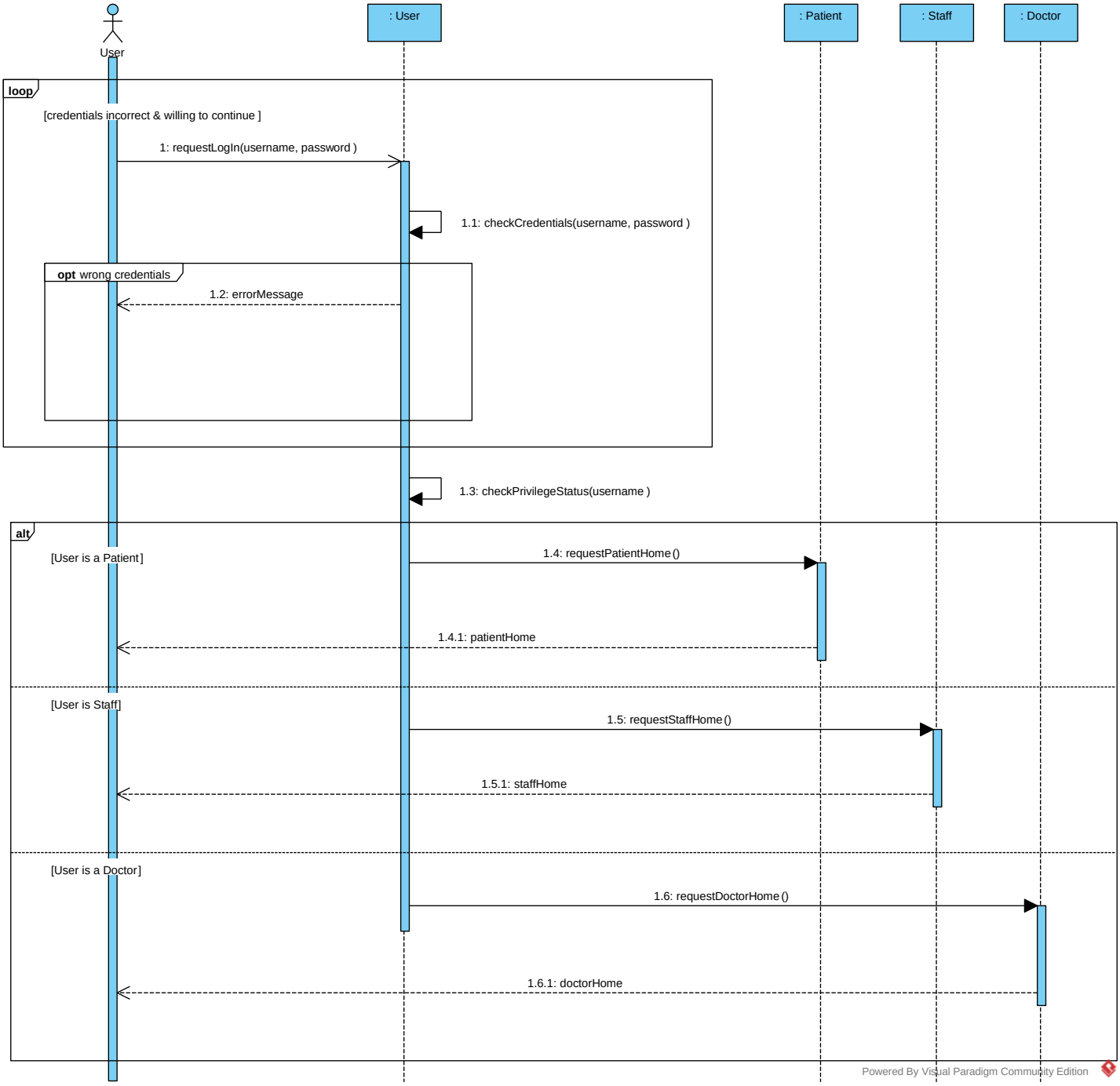
System Diagram: View Schedule



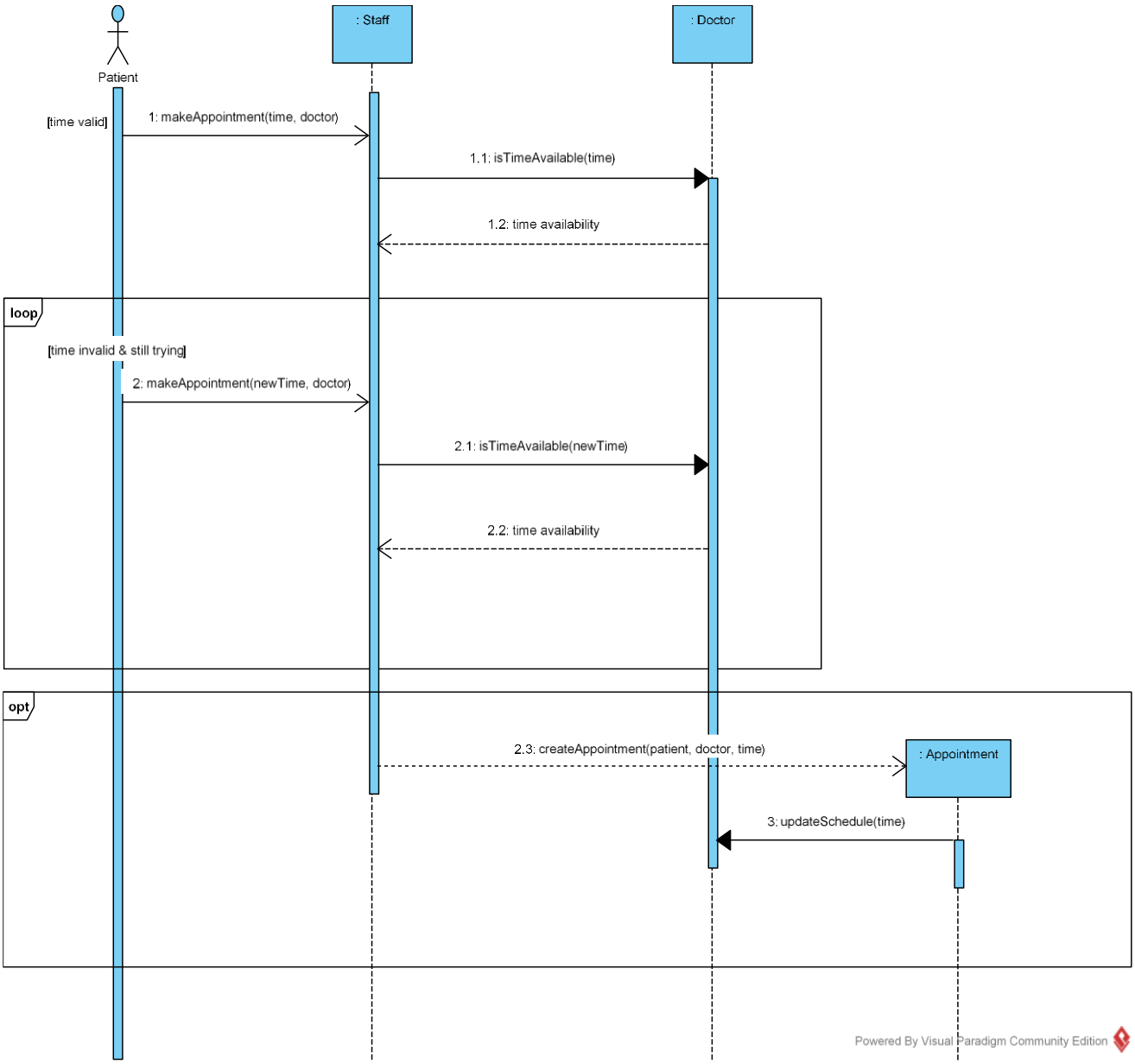
Sequence Diagram: Initial Visit

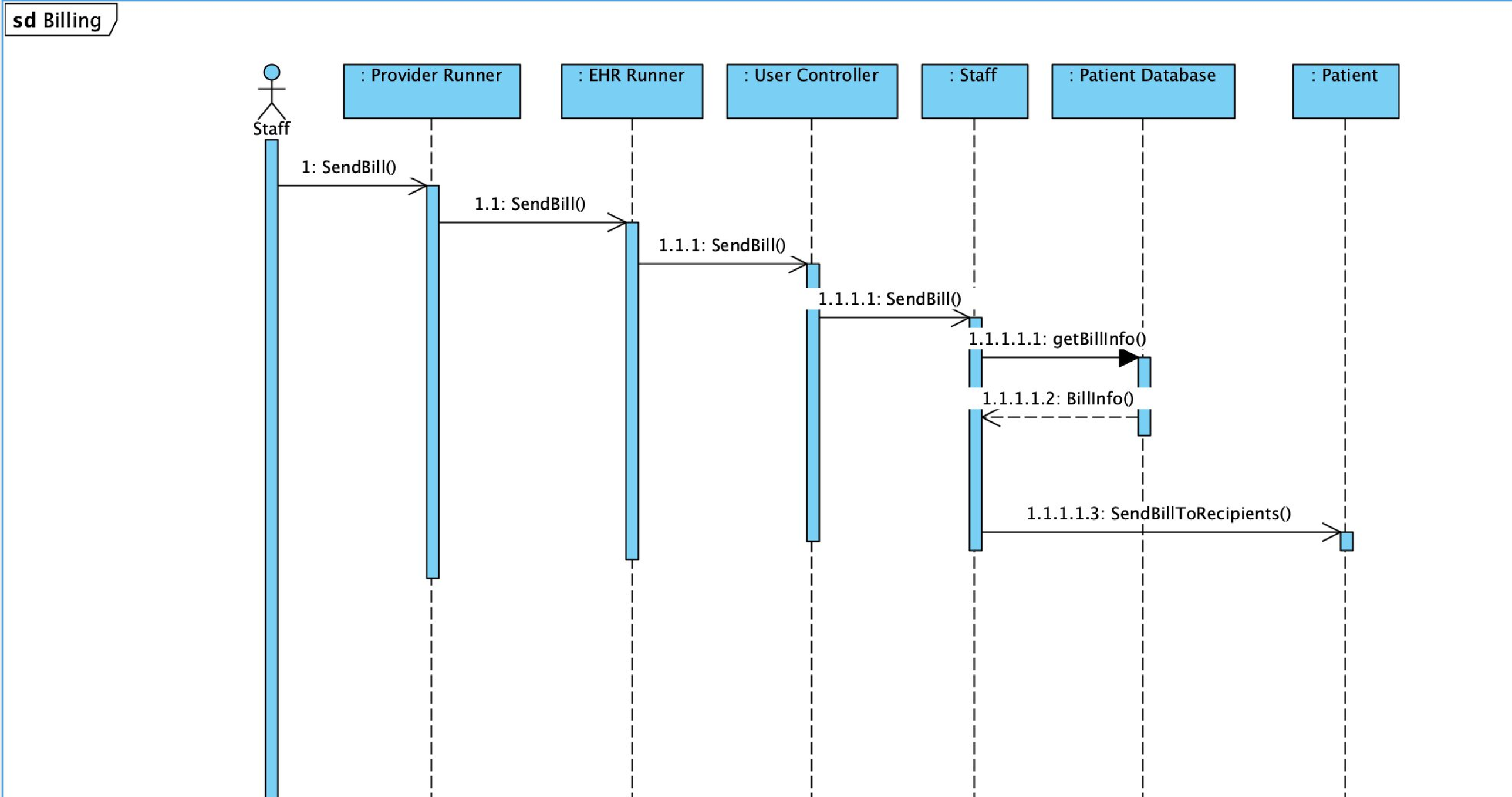


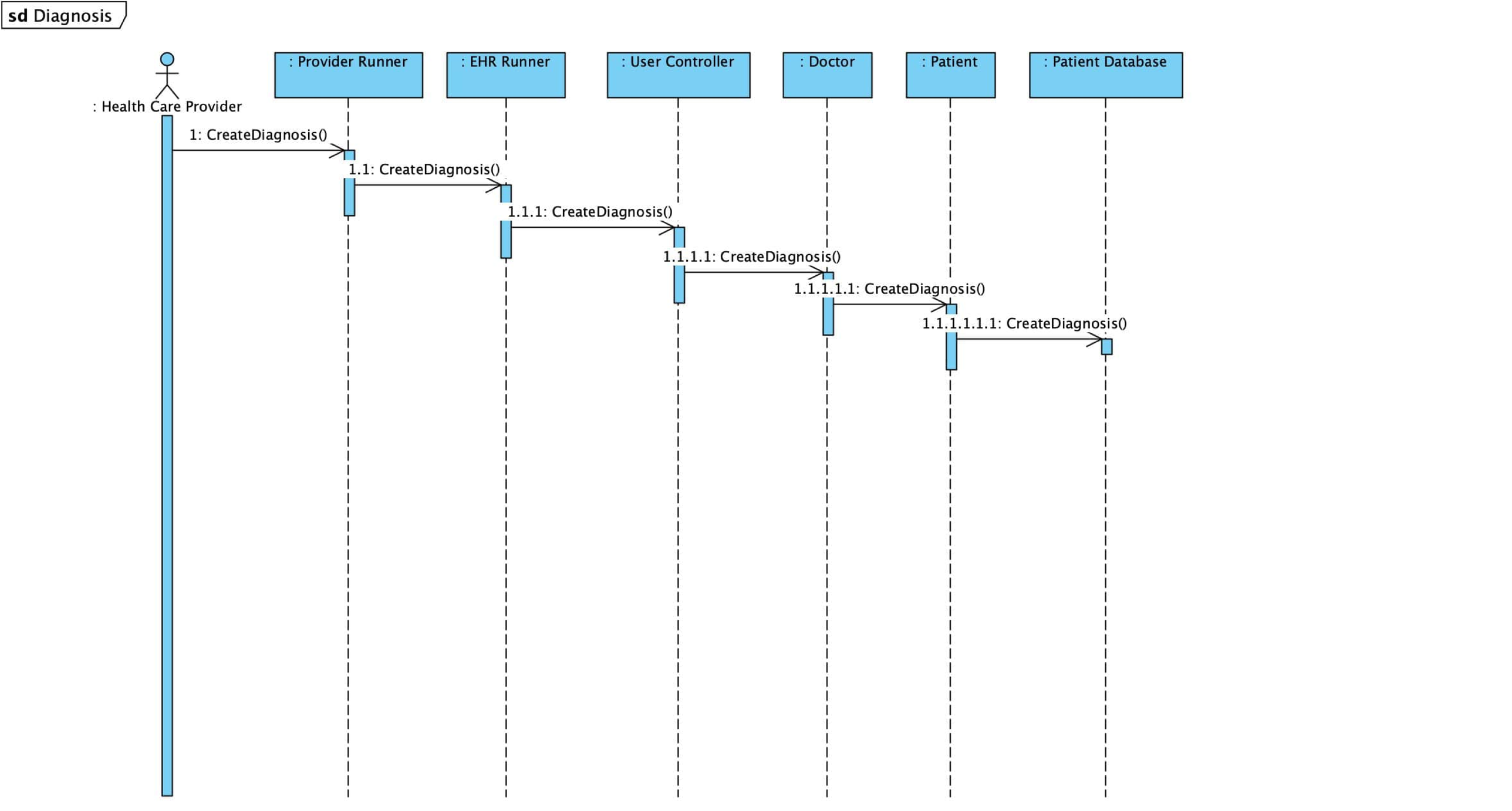
Sequence Diagram: Login

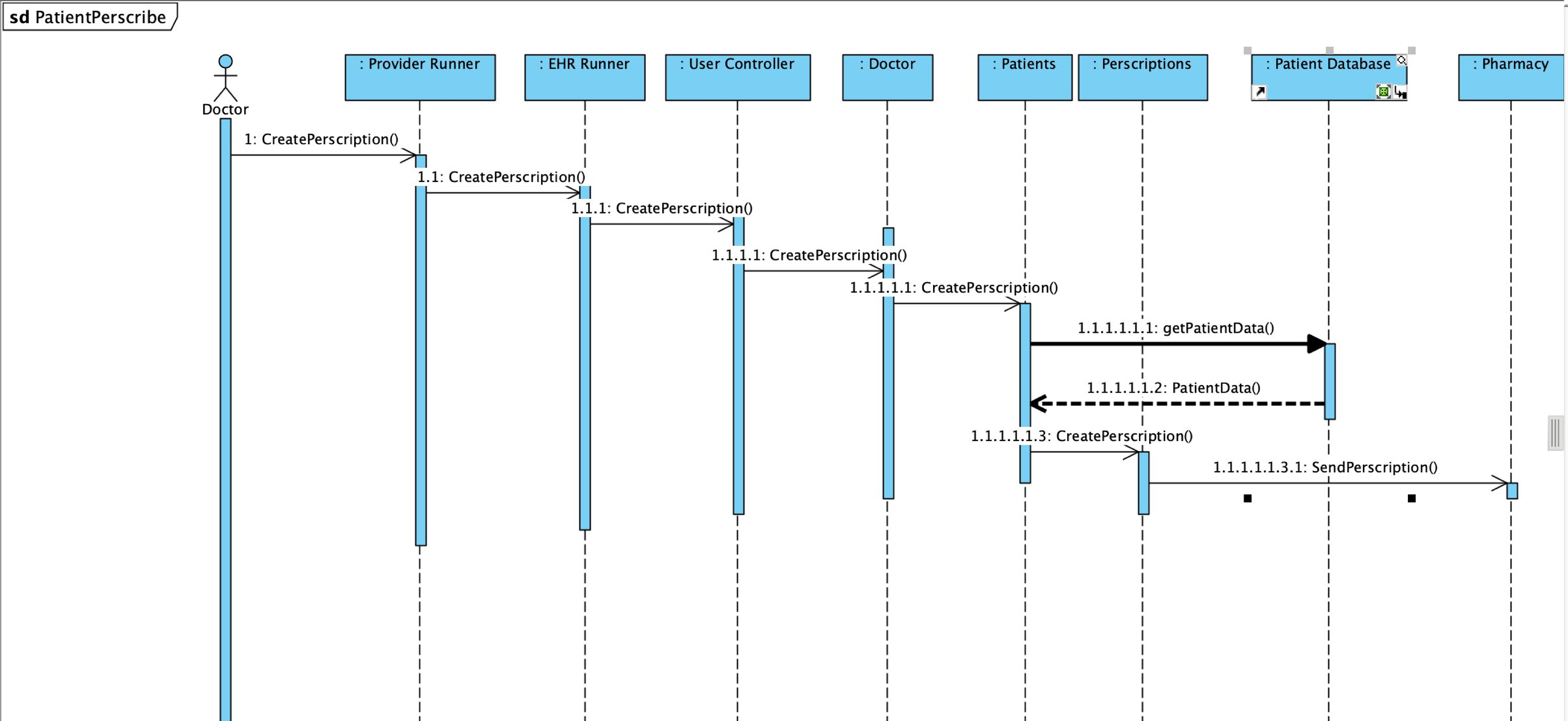


Sequence Diagram: make appointment

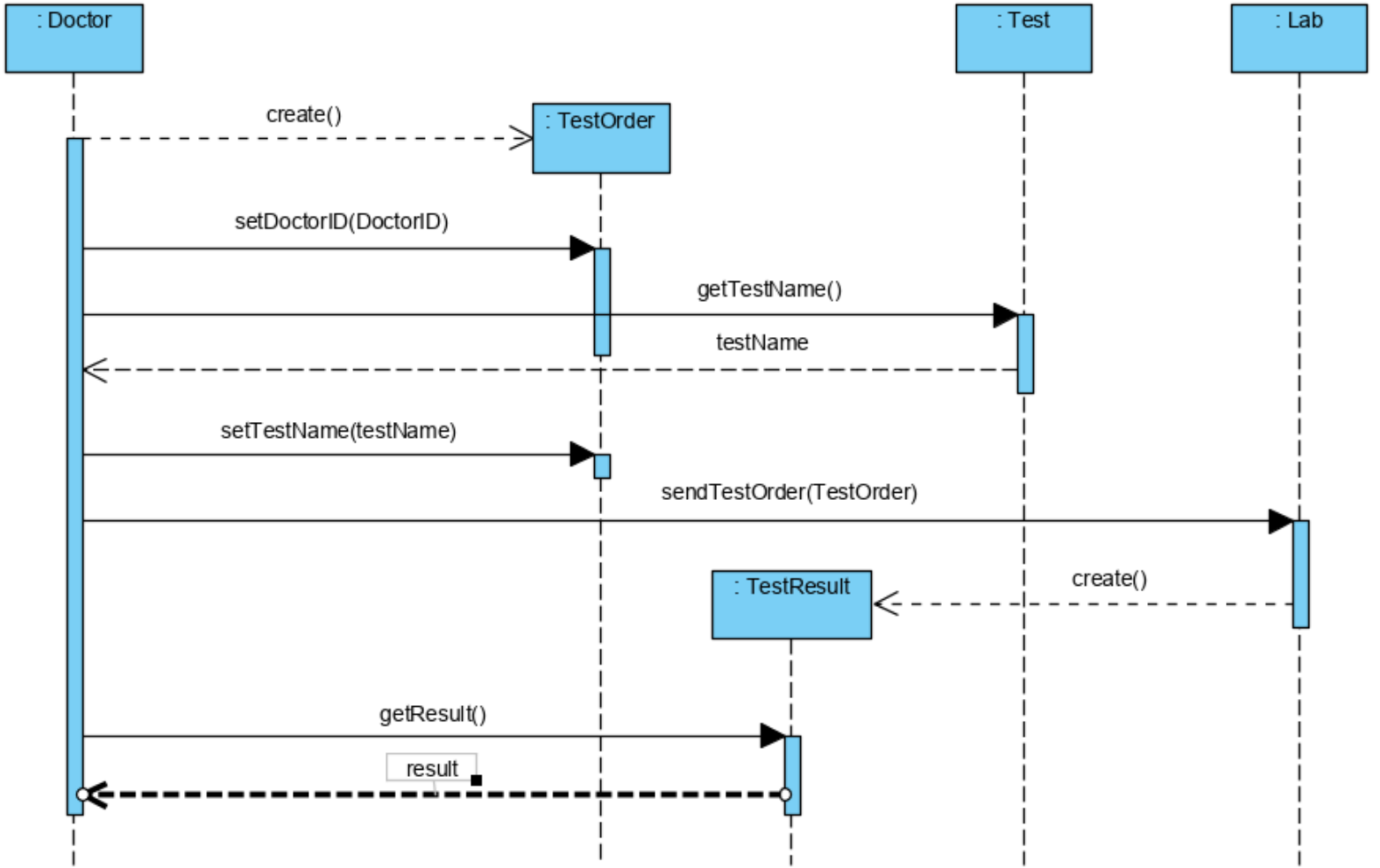


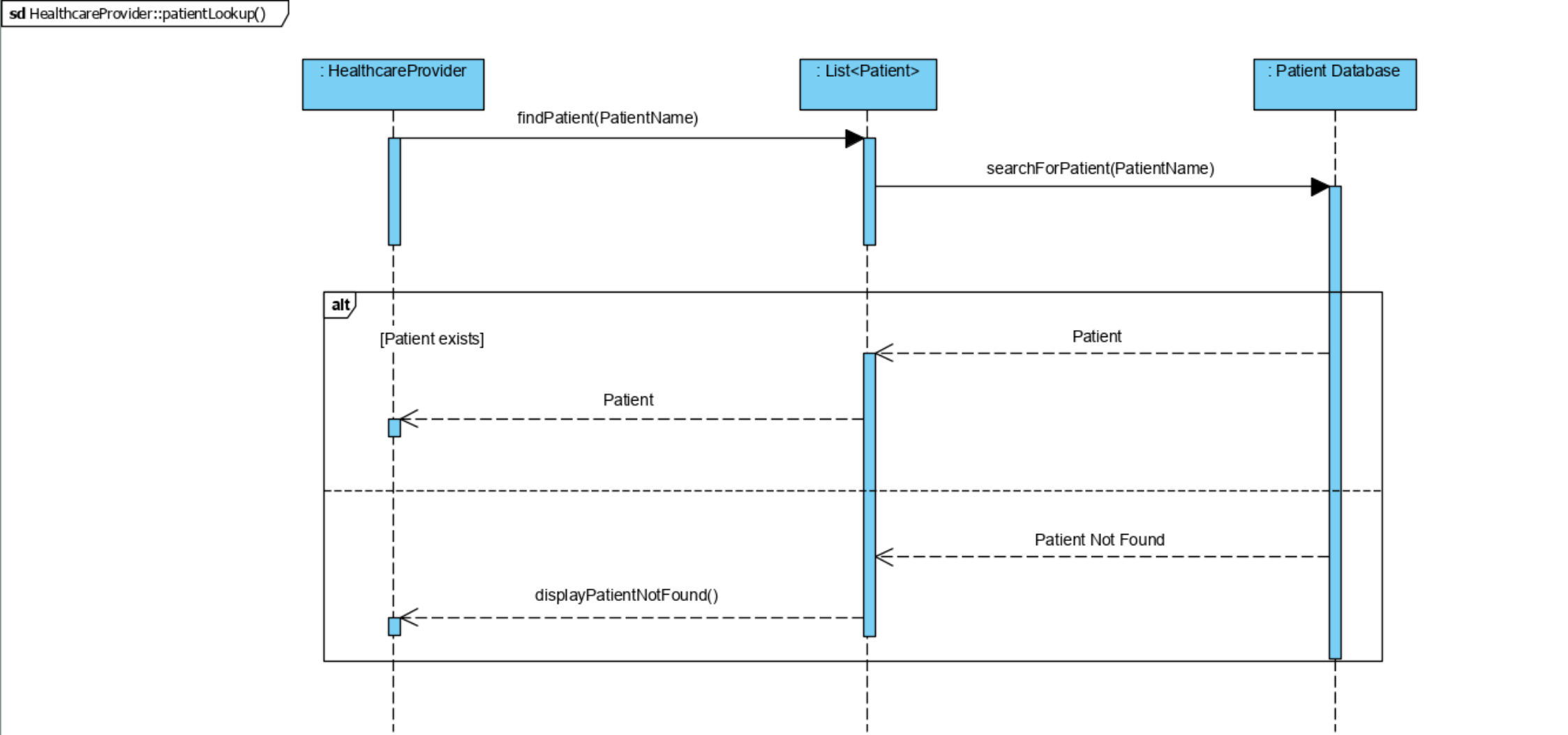

sd Billing
Staff
: Provider Runner
: EHR Runner
: User Controller
: Staff
: Patient Database
: Patient
1: SendBill()
1.1: SendBill()
1.1.1: SendBill()
1.1.1.1: SendBill()
1.1.1.1.1: getBillInfo()
1.1.1.1.2: BillInfo()
1.1.1.1.3: SendBillToRecipients()

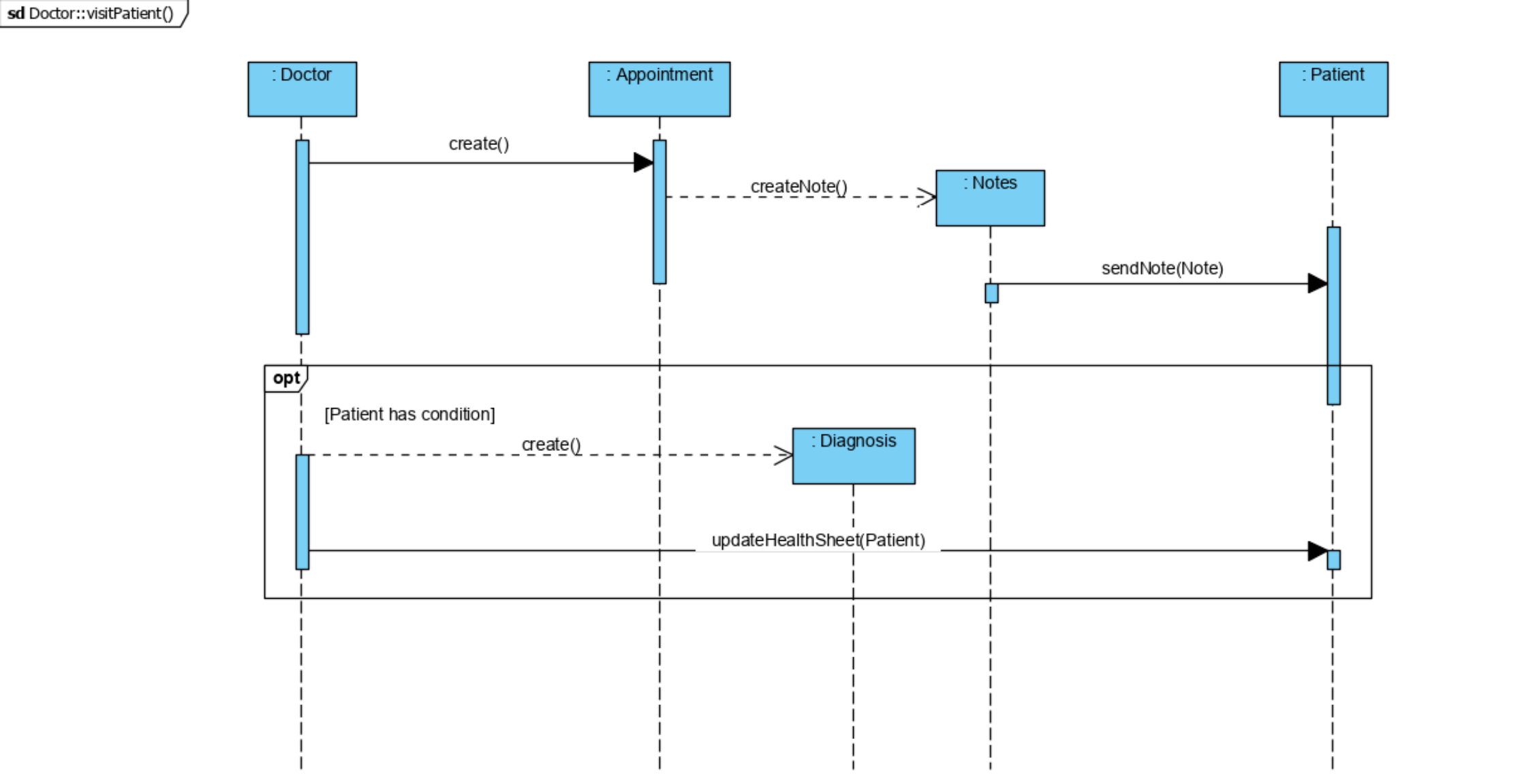

sd Diagnosis
: Health Care Provider
: Provider Runner
: EHR Runner
: User Controller
: Doctor
: Patient
: Patient Database
1: CreateDiagnosis()
1.1: CreateDiagnosis()
1.1.1: CreateDiagnosis()
1.1.1.1: CreateDiagnosis()
1.1.1.1.1: CreateDiagnosis()
1.1.1.1.1.1: CreateDiagnosis()


sd PatientPerscribe
Doctor
: Provider Runner
: EHR Runner
: User Controller
: Doctor
: Patients
: Perscriptions
: Patient Database
: Pharmacy
1: CreatePerscription()
1.1: CreatePerscription()
1.1.1: CreatePerscription()
1.1.1.1: CreatePerscription()
1.1.1.1.1: CreatePerscription()
1.1.1.1.1.1: getPatientData()
1.1.1.1.1.2: PatientData()
1.1.1.1.1.3: CreatePerscription()
1.1.1.1.1.3.1: SendPerscription()

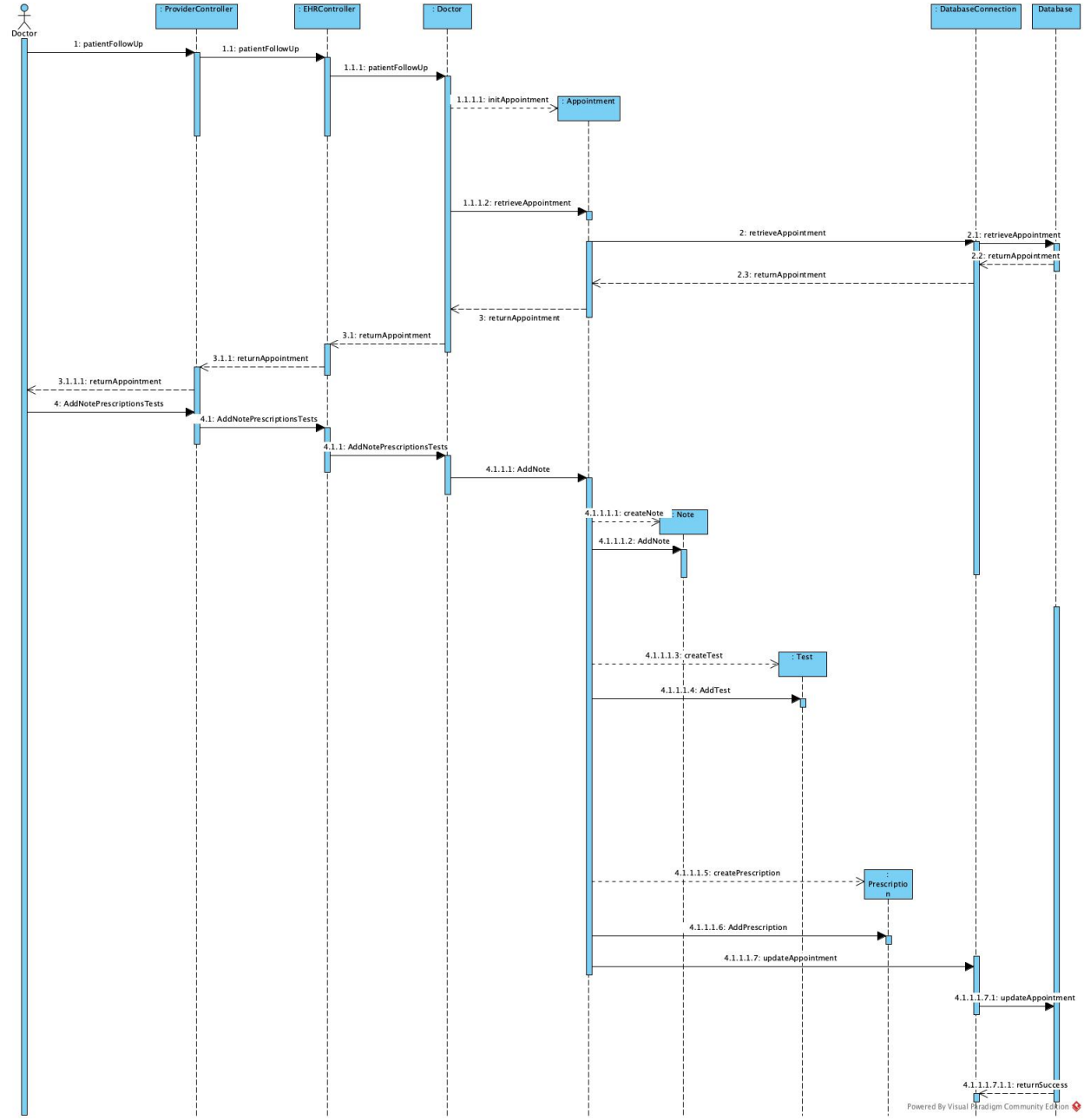
sd Doctor::orderTest()



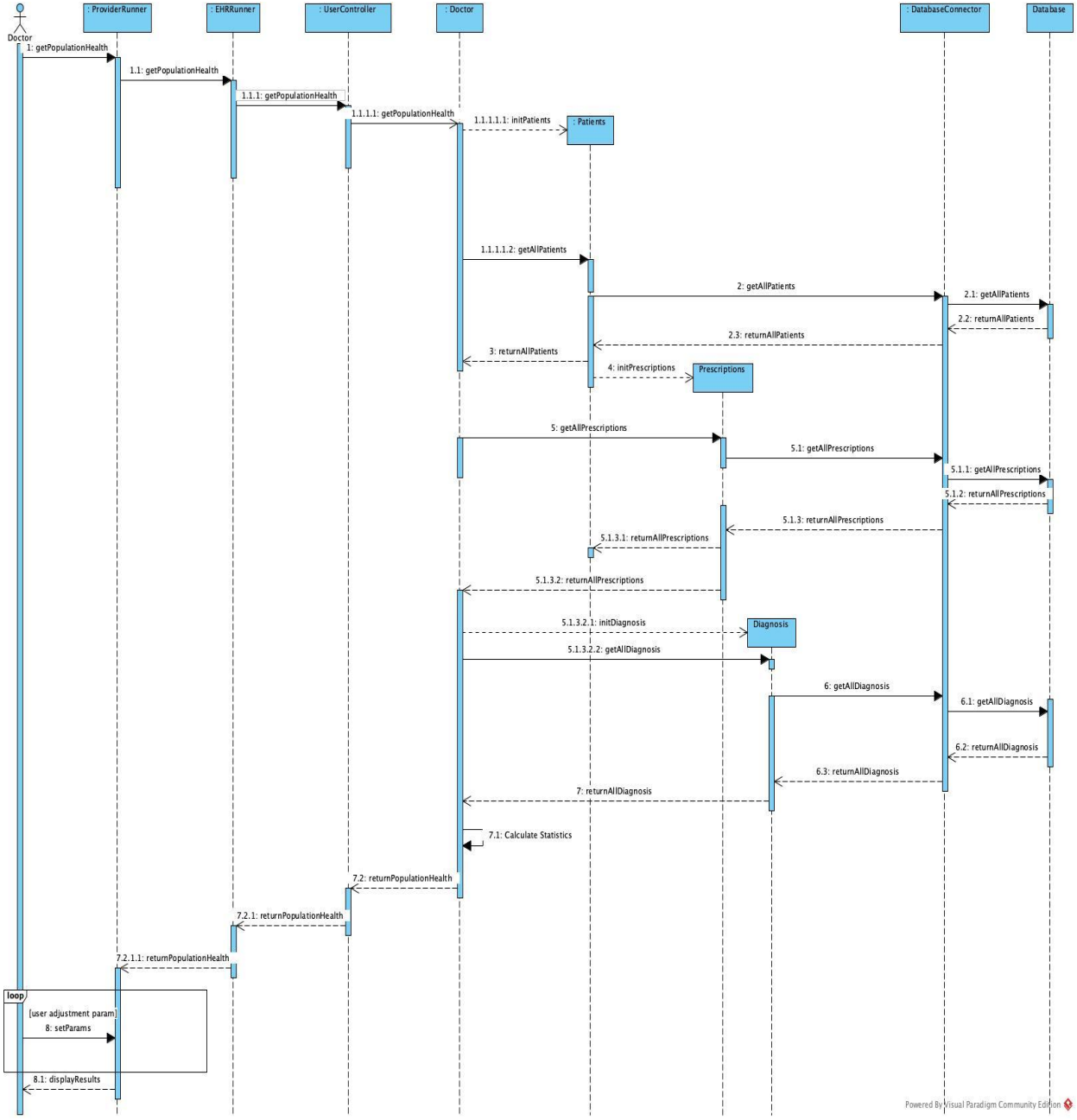

sd HealthcareProvider::patientLookup()
: HealthcareProvider
: List<Patient>
: Patient Database
findPatient(PatientName)
searchForPatient(PatientName)
alt
[Patient exists]
Patient
Patient
Patient Not Found
displayPatientNotFound()


sd Doctor::visitPatient()
: Doctor
: Appointment
: Patient
create()
createNote()
: Notes
sendNote(Note)
opt
[Patient has condition]
create()
: Diagnosis
updateHealthSheet(Patient)

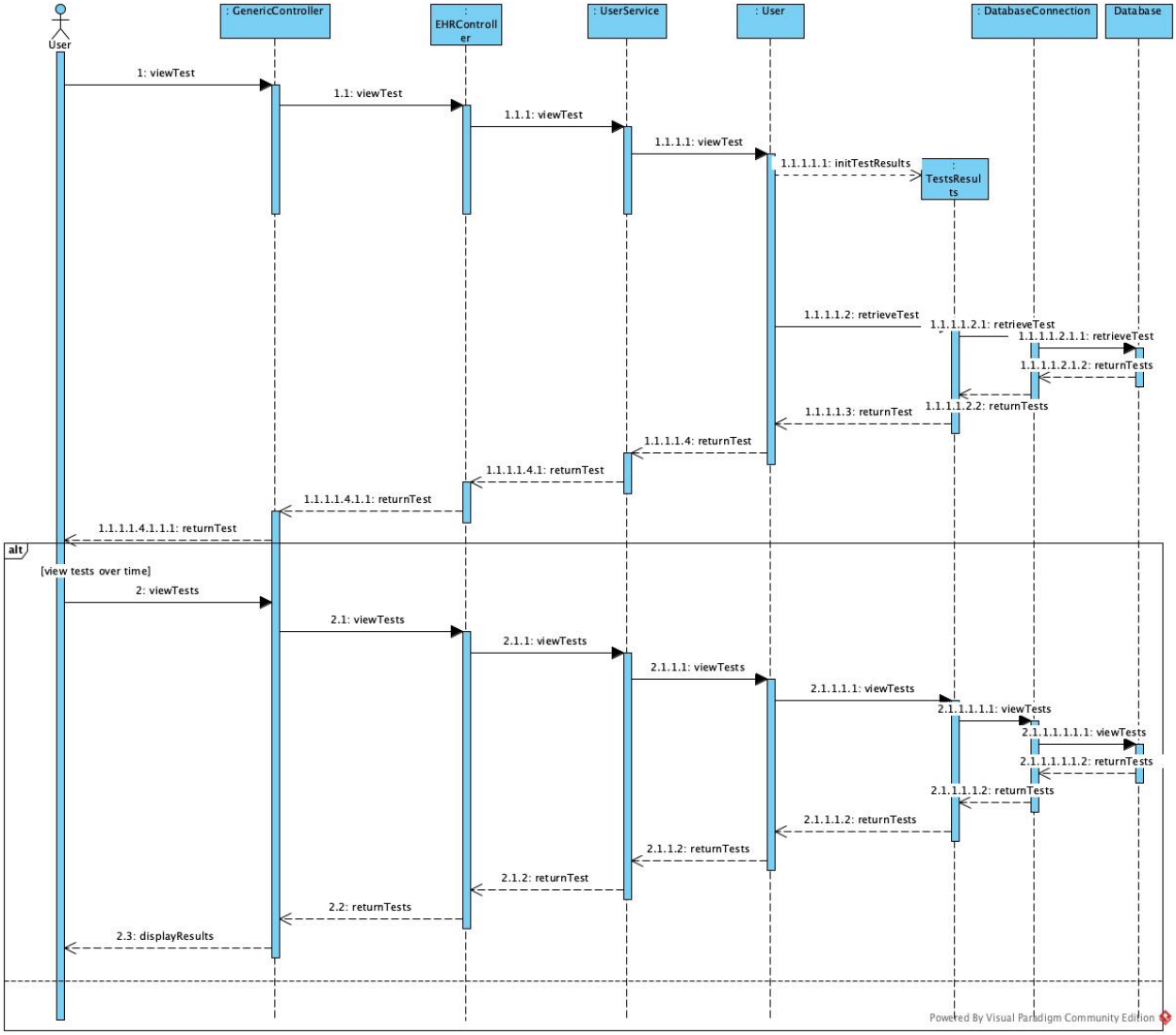
Sequence Diagram: Patient Follow-Up



Sequence Diagram: Population Health



Sequence Diagram: View Tests



TEST CASES

CShare Iteration 2

Log In

Information

Primary Actors Users

Details

Level	User Goal
Requirement	REQ011, REQ007
Success Criteria	User is able to log in and is directed to their home screen. Home screen is different for each user type. Users must only see the home screen for their specific type. Invalid credentials should not allow the user to view any home screen.
Failure Criteria	After providing correct credentials, user is not redirected to home screen. User is directed to an inappropriate home screen for their type. Invalid credentials allow a user to view a home screen.
Test Data	Fake users of all types (Doctor, Staff, Patient) with log in credentials. A set of log in credentials not belonging to any user.
Preconditions	Fake users are loaded into the testing database
Postconditions	Fake users are removed from the testing database

Test Execution Steps

Main Execution Scenario

1. The doctors user credential is entered in
2. System validates the user credentials and redirects to proper location
3. Test suite checks to ensure that system directed to the doctor view
4. Test suite logs out doctor from system
5. The staff's user credential is entered in
6. System validates the users credentials and redirects to proper location
7. Test suite checks to ensure that system directed to the staff view
8. Test suite logs out staff from system
9. The patients users credential is entered in
10. System validates the patients credentials and redirects to proper location
11. Test suite checks to ensure that system directed to the patient's view
12. Test suite logs out patient from system
13. Test suite enters invalid credentials
14. System validates credentials, recognizes they are invalid, and prints error message

Extensions

2a. The system cannot validate the doctor.

1. System prints error message
2. Test case fails

6a. The system cannot validate the staff.

1. System prints error message
2. Test case fails

10a. The system cannot validate the patient

1. System prints error message
2. Test case fails

14a. The system validates the credentials and redirects to landing page

1. Test case fails

Doctor Appointment Schedule

Information

Primary Actors Doctors, Staff

Details

Level	User Goal
Requirement	REQ005, REQ002, REQ003
Success Criteria	The doctor has a time sheet that updates when new appointments are created, retains all previous appointment information, and is viewable. Appointments cannot be created in time slots where existing appointments have already been created. Appointments must include both a patient and a doctor as registered attendees.
Failure Criteria	Created appointments are not visible to the doctor. Appointments can be created in the same time slots as previously schedule appointments. Appointments only include a doctor only.
Test Data	Fake users of all types (Doctor, Staff, Patient) with log in credentials. An appointment loaded to the database for a doctor and a patient.
Preconditions	Fake users are loaded into the testing database, fake appointments are loaded to database
Postconditions	Fake users and fake appointment are removed from the testing database

Test Execution Steps

Main Execution Scenario

1. The test suite checks to see all appointments for a doctor
2. The system returns all appointments for the doctor
3. Test suite tries to create new valid appointment between a doctor and a patient
4. System stores the appointment
5. Test suite checks to see all appointments for a doctor
6. System returns all appointments for the doctor
7. Test suite duplicates previously created valid appointment between doctor and patient
8. System rejects appointment and prints error message
9. The test suite checks to see all appointments for a doctor
10. The system returns all appointments for the doctor
11. Test suite then tries to create an appointment with only a doctor
12. System rejects appointment and prints error message
13. The test suite checks to see all appointments for a doctor
14. The system returns all appointments for the doctor

Extensions

2a. The system doesn't return any appointment
    1. Test case fails

4a. System returns an error message and does not store the appointment
    1. Test case fails

6a. System does not return both previously created appointments and new appointment

1. Test case fails

8a. System does not reject the new appointment

1. Test case fails

9a. System does not return all of the appointments

1. Test case fails

12a. System does not reject appointment and creates and stores the appointment

1. Test case fails

14a. System does not return all appointments

1. Test case fails

Patient Appointment Schedule

Information

Primary Actors Patients

Details

Level	User Goal
Requirement	REQ005, REQ007
Success Criteria	The patient can view previously created appointments.
Failure Criteria	Created appointments are not visible to the patient.
Test Data	Fake users of all types (Doctor, Staff, Patient) with log in credentials. An appointment loaded to the database for a doctor and a patient.
Preconditions	Fake users are loaded into the testing database, fake appointments are loaded to database
Postconditions	Fake users and fake appointment are removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite checks all appointments for a patient
2. System returns all appointments for a patient
3. Test suite adds a new valid appointment for the patient
4. System adds the new appointment
5. Test suite asks checks for all appointments for a patient
6. System returns all appointments

Extensions

2a. Not all appointments are returned
    1. Test fails

4a. System returns an error message when trying to add the new appointment
    1. Test fails

6a. Not all appointments are returned
    1. Test fails

Patient Information

Information

Primary Actors Health Care Providers, Patients

Details

Level	User Goal
Requirement	REQ001, REQ010, REQ011, REQ013
Success Criteria	All patient information is returned and is correct
Failure Criteria	Patient information returned is incomplete or incorrect
Test Data	Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions	All test data is loaded into testing database
Postconditions	All test data is removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite asks for personal information regarding one patient as a doctor
2. System returns patient information
3. Test suite asks for diagnoses, prescriptions, and appointments for a patient
4. System returns this information
5. Test suite asks for a different patients information and diagnoses, prescriptions, and appointments.
6. System returns information
7. Test suite checks to ensure that this information returned differs from other patient information
8. Test suite asks for personal information of a patient as a patient
9. System returns error

Extensions

2a. System cannot find patient and returns error
    1. Test suite fails

4a. System cannot find this information or information is incorrect
    1. Test suite fails

7a. System cannot find information or information is incorrect
    1. Test suite fails

9a. System returns information
    1. Test suite fails

Doctor Information

Information
Primary Actors Doctor

Details
Level User Goal
Requirement REQ004, REQ012
Success Criteria All doctor information is accessible and visible
Failure Criteria Doctor information is either inaccessible or not visible.
Test Data Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions All test data is loaded into testing database
Postconditions All test data is removed from the testing database

Test Execution Steps
Main Execution Scenario

1. Test suite asks for doctor information
2. System returns doctor information
3. Test suite adds new patient information and assigns to doctor
4. System updates information
5. Test suite asks for doctor information
6. System returns updated information

Extension

2a. System can't find or returns incorrect doctor information
    1. Test case fails
4a. System cannot update information
    1. Test case fails
6a. System returns incorrect information
    1. Test case fails

Clinical Tests

Information

Primary Actors Doctor, Patients, Staff

Details

Level	User Goal
Requirement	REQ008, REQ012
Success Criteria	Doctors and staff can see clinical tests for all of their patients, patients can see their tests. New tests can be ordered.
Failure Criteria	Tests are not visible for any user. Patients can view other patients tests. New tests cannot be ordered.
Test Data	Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions	All test data is loaded into testing database
Postconditions	All test data is removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite asks system for all ordered tests for all of a doctor's patients as a doctor
2. System returns all ordered tests for all of a doctor's patient
3. Test suite asks system for all ordered tests for a patient as a patient
4. System returns all ordered tests for the patient
5. Test suite tries to create a new test for a patient
6. System creates the new test for the patient
7. Test suite tries to access all patient tests as a patient
8. System returns an error

Extensions

2a/4a. System can't find tests or doesn't return all tests
    1. Test case fails

6a. System returns error message and does not create the new test
    1. Test case fails

8a. System returns all tests
    1. Test case fails

Orders

Information

Primary Actors Doctor, Patients, Staff

Details

Level	User Goal
Requirement	REQ009
Success Criteria	Orders and prescriptions are visible to doctors and patients. Orders and prescriptions can be created by doctors.
Failure Criteria	Orders and prescriptions are not visible or cannot be created
Test Data	Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions	All test data is loaded into testing database
Postconditions	All test data is removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite asks for all orders and prescriptions for a particular patient as a doctor
2. System returns information
3. Test suite asks for all orders and prescriptions for a particular patient as a patient
4. System returns all information
5. Test suite tries to create an order as a doctor
6. System creates order
7. Test suite tries to create a prescription as a doctor
8. System creates prescription
9. Test suite tries to create prescription as a patient
10. System rejects prescription
11. Test suite tries to create an order as a patient
12. System rejects order

Extensions

2a/4a. System cannot find or returns incorrect information

1. Test case fails

5a/7a. System cannot create prescription or order

1. Test case fails

10a/12a. System access order or prescription and creates it

1. Test case fails

File Sharing

Information

Primary Actors Doctor, Staff

Details

Level	User Goal
Requirement	REQ014
Success Criteria	Patient information is securely stored to an encrypted file.
Failure Criteria	Patient information is not securely stored to encrypted file.
Test Data	Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions	All test data is loaded into testing database
Postconditions	All test data is removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite asks for patient information dump
2. System returns patient information dump

Extensions

2a. System returns an error or the file is not created properly

1. Test suite fails

Population Health

Information

Primary Actors Doctor, Staff

Details

Level	User Goal
Requirement	REQ015
Success Criteria	Population health information is correctly calculated
Failure Criteria	Population health information is not correctly calculated
Test Data	Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions	All test data is loaded into testing database. Population health information is predetermined for test data.
Postconditions	All test data is removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite asks for population health parameters
2. System returns calculated population health parameters
3. Test suite compares returned population health parameters to previously determined values.

Extensions

2a. System returns an error when calculating parameters
    1. Test suite fails

3a. Returned parameters don't match the previously created population health parameters.
    1. Test suite fails.

Insurance Coding

Information

Primary Actors Doctor, Staff, Patient

Details

Level	System Goal
Requirement	REQ006, REQ013, REQ001
Success Criteria	Each condition is coded with a specific code
Failure Criteria	Each condition is not coded with a specific code
Test Data	Multiple fake patients with tests, diagnosis, prescriptions, and appointments. Fake doctors and staff.
Preconditions	All test data is loaded into testing database. Billing codes predetermined
Postconditions	All test data is removed from the testing database

Test Execution Steps

Main Execution Scenario

1. Test suite enters different conditions for a patient
2. System updates patient
3. Test suite asks for billing codes for the patient
4. System returns codes

Extensions

2a. System does not update patient.

1. Test case fails.

4a. System returns incorrectly coded billing information.

1. Test case fails.

	Login	Doctor Schedule	Patient Schedule	Patient Information	Doctor Information	Clinical Tests	Orders	File Sharing	Population Health	Insurance Co
REQ001				X						X
REQ002		X								
REQ003		X								
REQ004					X					
REQ005		X	X							
REQ006										X
REQ007	X		X							
REQ008						X				
REQ009							X			
REQ010				X						
REQ011	X			X						
REQ012					X	X				
REQ013				X						X
REQ014								X		
REQ015									X	

GITHUB: https://github.com/shenoisam/Software1/issues

Timecard Report: https://shenoisam.github.io/Software1/

Linked Issue tracking system: https://github.com/shenoisam/Software1/issues

MAVEN demo: available within code on github

Suggested Point Redistribution:

> At the moment we feel no need to redistribute the points as the group members have been contributing evenly.