Final report

Medical Information System

INSY 5341 Analysis and Design

Due May 4, 2021 at 11:59pm

Project Group 3:

Garasky, Kimberly

Ji, Wenhan

Li, Yueqi

Sassa, Dilson

Voronov, Denis

Wan, Yuting

Introduction

Presented here is the final project report for the Medical Information System by Group 3. This report is divided into four sections: planning, analysis, design and implementation.

The planning section includes the introduction about the problem domain, System Request, Feasibility analysis, and Requirements Definition.

The analysis section includes the activity diagram, use case diagram, use case descriptions, initial class diagram, and sequence diagrams.

The design section includes the final class diagram, package diagram, database design, and data access and manipulation design.

The implementation section includes sample screen shots of the software.

Contents

1. Planning

1.1 Introduction about the problem domain

For many years one of the most serious problems for the United States has been a dramatic shortage of medical employees. As the country's population is growing and getting older, staff shortage becomes more and more significant. One of the ways to solve this problem is to make hospitals and healthcare workers more effective. Here is where the usage of the latest Information Technologies can help considerably.

Medical staff is usually spending a lot of time on paperwork - medical tests, health history, referrals, prescriptions, bills, insurance claims, financial reports etc. Time spent on this routine could be spent on serving a larger number of patients and all this can make medicine more affordable.

Right now there are many ERP systems that can automate most of the tasks, and some of them are designed specifically for healthcare. If in the past a hospital could utilise different systems for appointments, billing, scheduling, financial reporting - the latest trends show that hospitals tend to change them to a many-in-one system that unites all these functions and makes them compatible. Unfortunately, such systems are quite complicated for non-IT professionals, very inflexible and expensive both in upfront costs and aftermarket support.

The system we are going to design is planned as a full-blown software package, consisting of the core mandatory for every organization and add-in modules that can be installed separately on demand. It will be convenient, have a clear and intuitive interface and affordable for small and mid size hospitals. The first version of the system will include only the core features.

1.2 System Request

System Request: Medical Information System

Project sponsor: John Smith, Chief Technology Officer

Business Need: This System is developed for Hospital organizations and is aimed at serving non-IT professionals in the Medical field.

Business requirements

The following are the primary users of this system: Administrators, Medical staff (Nurses, Medical doctors, Pharmacists), Accounting Department, Hospitals Human Resource managers, Top Management

- Appointments and schedule management
- Non-IT professionals will be able to follow up on patients progress reports
- Nurses will be able to use the Cloud based system to store patients records
- MDs & DOs will be able to get speciality doctors referrals automatically
- Accounting Department will get accurate information on services rendered and avoid errors in medical bills
- Hospital Human Resources managers will be able to hire medical staff regardless of their IT capabilities
- Top Management will be able to track the financial results of a hospital

Business value

A considerable amount of time is spent on medical records and medical billing by medical staff. This new system will reduce time consuming non-medical tasks by more than 30% which in turn, will save hospitals money. Integrating schedule appointments and reminders, prescriptions and medical tests orders into one system managed by a single medical staff.

As a result, hospitals will increase their capabilities to see more patients. Additional time will be allocated to primarily medical assignments and patient outcome will greatly improve as a result. More patients and less paperwork will increase the bottom line.

Special Issues or Constraints

- Merging the new system to replace the current system will take considerable time. Within 2 years of implementation the cost/benefit analysis will be apparent.
- Issues with Cloud based systems might occasionally arise that will require additional service.

1.3 Feasibility Analysis

Technical Feasibility

The Medical Information System is feasible technically, although there is some risk.

The project risks regarding familiarity with hospital processes is medium

- The project doesn't introduce new business processes, but only makes the existing ones easier to manage in the digital format
- Some of the processes that will not be integrated into the first version of the system will have to be reconciled with the current format of documentation
- The project group will pay special attention to the legal compliance of the system

The project risks regarding the technologies used is medium

- Cloud computing in medical information systems is quite common in 2021. For the last several years the IT community has developed a lot of knowledge of the domain and the requirements
- The preferred distribution model SaaS (software as a service) is widely spread among major vendors of ERP and the market is ready for them
- Data security risk is low, the project will utilize distributed storage with high level of redundancy, multiple nodes used and regular backup

The project risk regarding familiarity with the software used is low

- There are 5 developers included into the project team: all of them are proficient with MySQL, HTML, CSS and Javascript, and 2 of them are experts in OpenStack and SpringBoot
- In case the project team faces technical difficulties (which is unlikely) and delays in work (medium possibility), they can attract third party experts as needed. The project budget has provisions for that.
- Major development tools are open source and free to use

The project size is considered low risk

- The project team will consist of only 7 people and will last 6 months. It enables the team to work using SCRUM process
- It'll be necessary to involve targeted customers (medium and small hospitals), which can cause a little problem
- The first version of the system will be thoroughly tested for bugs and performance before it is integrated into a real hospital

The system compatibility must be excellent. There is a medium risk here.

- Previously used databases are compatible with MySQL
- It'll be challenging to integrate the legacy accounting and HR systems into the new project

- Interface of the new system is going to be user friendly and intuitive to the medical staff, that will help to avoid data entry errors from the very start

Economic Feasibility

The rough estimate of the project.
Cost estimator:

- System development costs

  Total personnel cost is $204,000, the cost of hardware equipment required by the system is $104,600, software cost (Windows copyright) is $5000.

In summary, the total development cost is $**313,600**.

- System operating costs

Assuming that the operating cycle of the system is 5 years, and the annual operating cost is: $42,500, the total operating costs for 5 years will be **$212,500**

The total cost of system development and operation is $**526,100**, an average of **$105,220** per year.

Benefit estimator:

- Direct savings: Reducing the use of office equipment, paper and postal services can save about **$14,000** per year.
- Unquantifiable income: The increase in business volume increases the direct economic benefits by approximately **$80,000** per year.

  The total economic effect will sum up to **$94,000** per year. It is expected that the economic effect will expand to **$130,000** per year in 5 years.

Organizational Feasibility

This risk level is low, because the platform used to design the medical system is a cloud server, which is a model that can easily and on-demand access to a shared configurable computing resource pool. A medical system based on cloud computing can improve operational efficiency.

If during the implementation of the project, a reasonable project schedule can be formulated, and the project manager can coordinate multiple staff, design a reasonable organization, and the staff can cooperate efficiently with each other, the project will progress smoothly.

In addition, the most important reason for the successful implementation of the project is that the project manager needs to pick the staff with the required expertise. Only the staff skilled in the technology will not waste time on the project, and will not spend extra time training newcomers .The efficiency will be greatly improved when the results are finally delivered.

1.4 Requirements Definition

Nonfunctional Requirements

1. Operational Requirements
   1.1. The medical information system will include information from the office patient database, which contains information about patients, such as name, age, gender, phone number, address,payment information and insurance details.
   1.2. The medical information system will keep a record of each patient's medical history. This will include illnesses, surgeries, medical tests, and medical conditions a patient has had and the date they occurred or were diagnosed.
   1.3. The medical information system will store orders for prescriptions and medical tests. Staff will be able to submit orders for prescriptions to the pharmacy through a prescription order form, and order for tests to specialty labs through a test order form.
   1.4. A calendar will be integrated into the medical information system so that appointments can be scheduled.
   1.5. The medical information system will maintain patient financial information, record the costs billed to the patient and the overall amount owed, and collect payment.
2. Performance Requirements
   The medical information system will provide a better daily operational system for both patient and medical staff. The convenient and efficient system will provide better, faster services and experiences during the medical and healthcare processes. The system will use the patient financial record and payment history to collect payments after the appointment.
3. Security Requirements
   In order to protect patient confidentiality, staff will need to log into the medical information system with a unique username and password through an internal VPN server. If a user is inactive for two minutes, the system will automatically save and log out.
4. Cultural and Political Requirements
   No special cultural and political requirements are anticipated.

Functional Requirements

1. Maintain patient information
   1.1. The medical information system needs a database that contains patient information, including patient ID, patient name, age, gender, phone number, address, payment information and insurance details.
   1.2. The medical information system will contain records of each patient's medical history, including any illnesses, surgeries, tests, or other medical conditions a patient may have as well as the date they occurred or were diagnosed.
   1.3. Staff will be able to update the information to the system as necessary, such as if a patient moves or changes insurance providers.
2. Place prescription orders

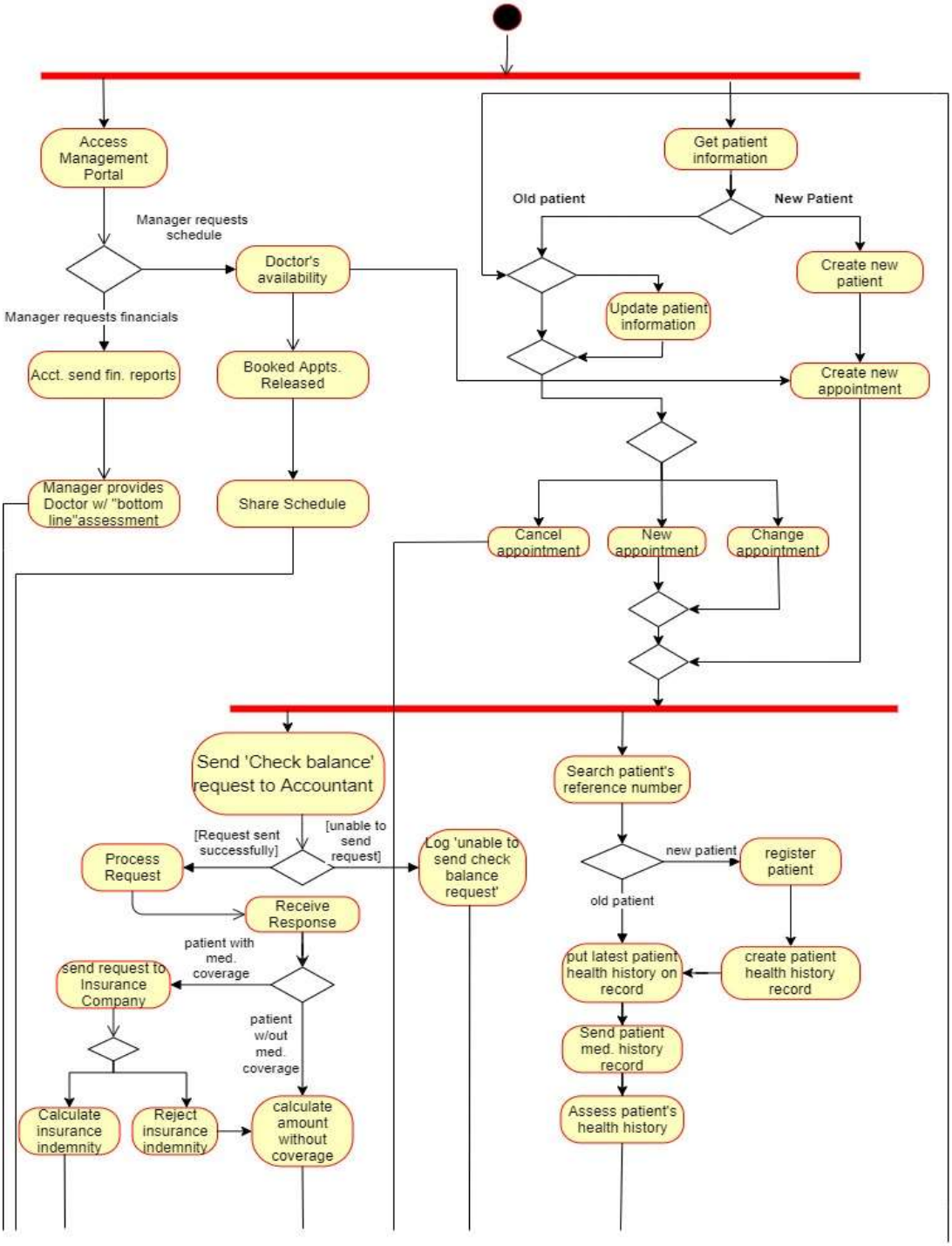
2.1. Staff will access the medical information system to order prescription drugs for patients. Staff will be able to search for a drug specifically by name.
2.2. To place the prescription drug order, staff will enter the patient's information as well as information about the pharmacy where the patient will pick up the prescription drug order. Staff will also include details regarding the order, such as the quantity of the drug and the directions for use.
2.3. The prescription order information will be sent to the patient's insurance (if applicable).
2.4. The prescription order information will be sent to the selected pharmacy to fill the order. The pharmacy will contact the patient directly when it is ready to be picked up. The pharmacy will collect the payment (if not covered by insurance) from the patient.
2.5. The system will track the orders with the patient information, then use the payment information history to collect bills.

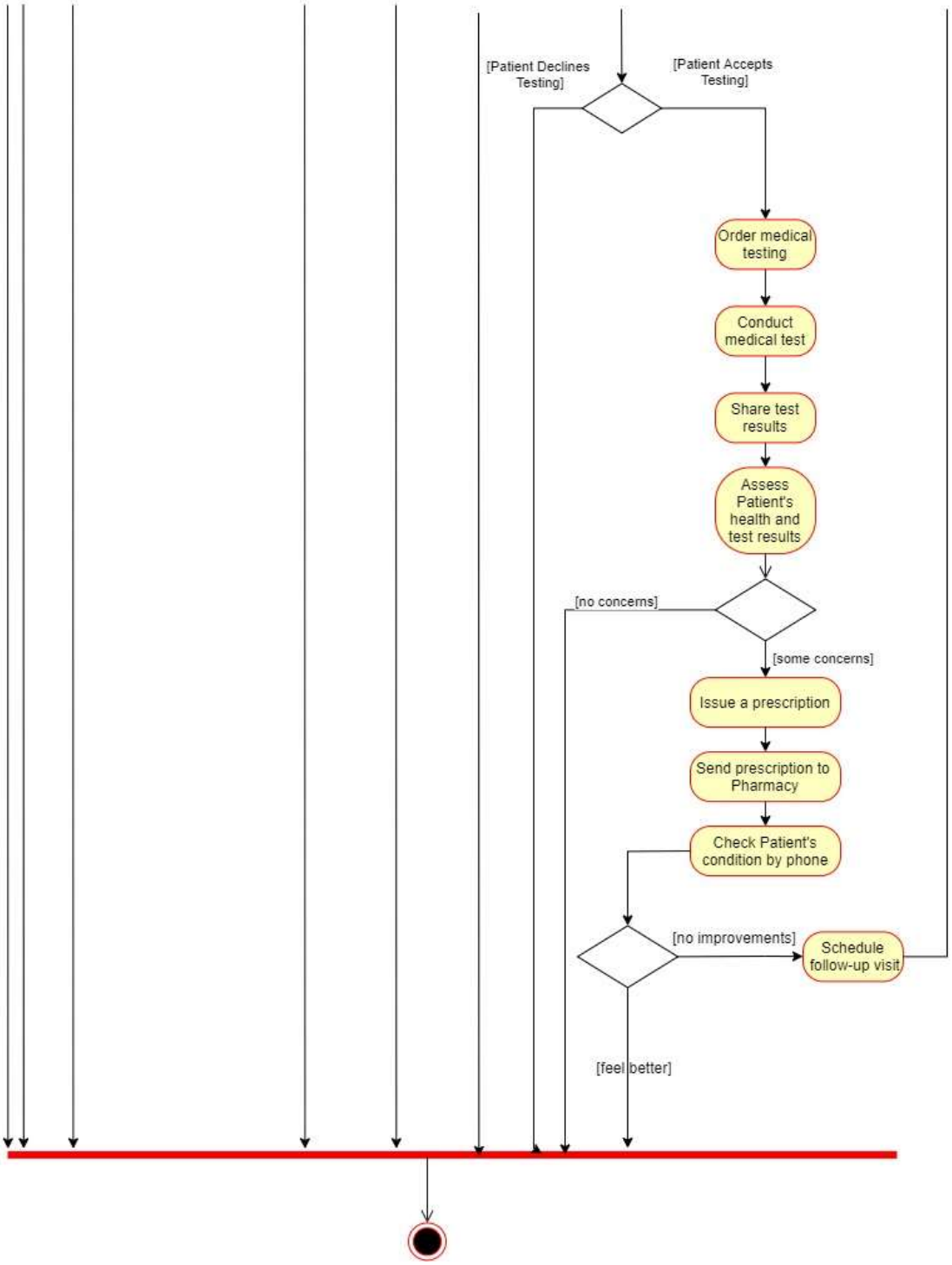
3. Place medical test orders
   3.1. Staff will access the medical information system to order medical tests for patients. Staff will be able to search for a test specifically by name.
   3.2. To place the test order, staff will enter the patient's information as well as information about the laboratory where the patient will go to have the test conducted. Staff will also include details regarding the order, such as the reasoning for the test and any special instructions (such as the patient should not eat for at least three hours beforehand).
   3.3. The test order information will be sent to the patient's insurance (if applicable).
   3.4. The test order information will be sent to the laboratory that will conduct the test. The patient will go to the laboratory at the appointed time. The laboratory will collect the payment (if not covered by insurance) from the patient.
   3.5. The laboratory will share the test results with the medical office. Staff will update the patient's medical history in the medical information system. Patients will be notified of their test results and staff will schedule follow up appointments if necessary.
   3.6. The medical information system will use patients' financial records to collect laboratory test fees.
4. Schedule appointments and reminders
   4.1. Staff will be able to schedule appointments in the medical information system for patients to visit the office to receive medical attention. Staff will include the patient's contact information, the date, time and reason for visit when the appointment is made.
   4.2. Staff will be able to make changes to appointments in the case that a patient needs to cancel or reschedule.
   4.3. The medical information system sends a notification to staff two days before a patient's appointment. At that time, staff will call the patient using the phone number listed in the system for that patient to remind them of the upcoming appointment.
5. Bill patients for services and collect payment
   5.1. Staff will be able to update patient's payment information to the system.
   5.2. Patients will receive a bill following their appointment with an itemized list of the amount owed for the services provided, including any tests and prescriptions administered.
   5.3. Patients will have 30 days to submit payment. If payment is not received by the due date, managers will have the option to sell the unpaid bills to a collections agency.

2. Analysis

2.1 Activity Diagram

This activity diagram portrays the primary activities and relationships in our system.



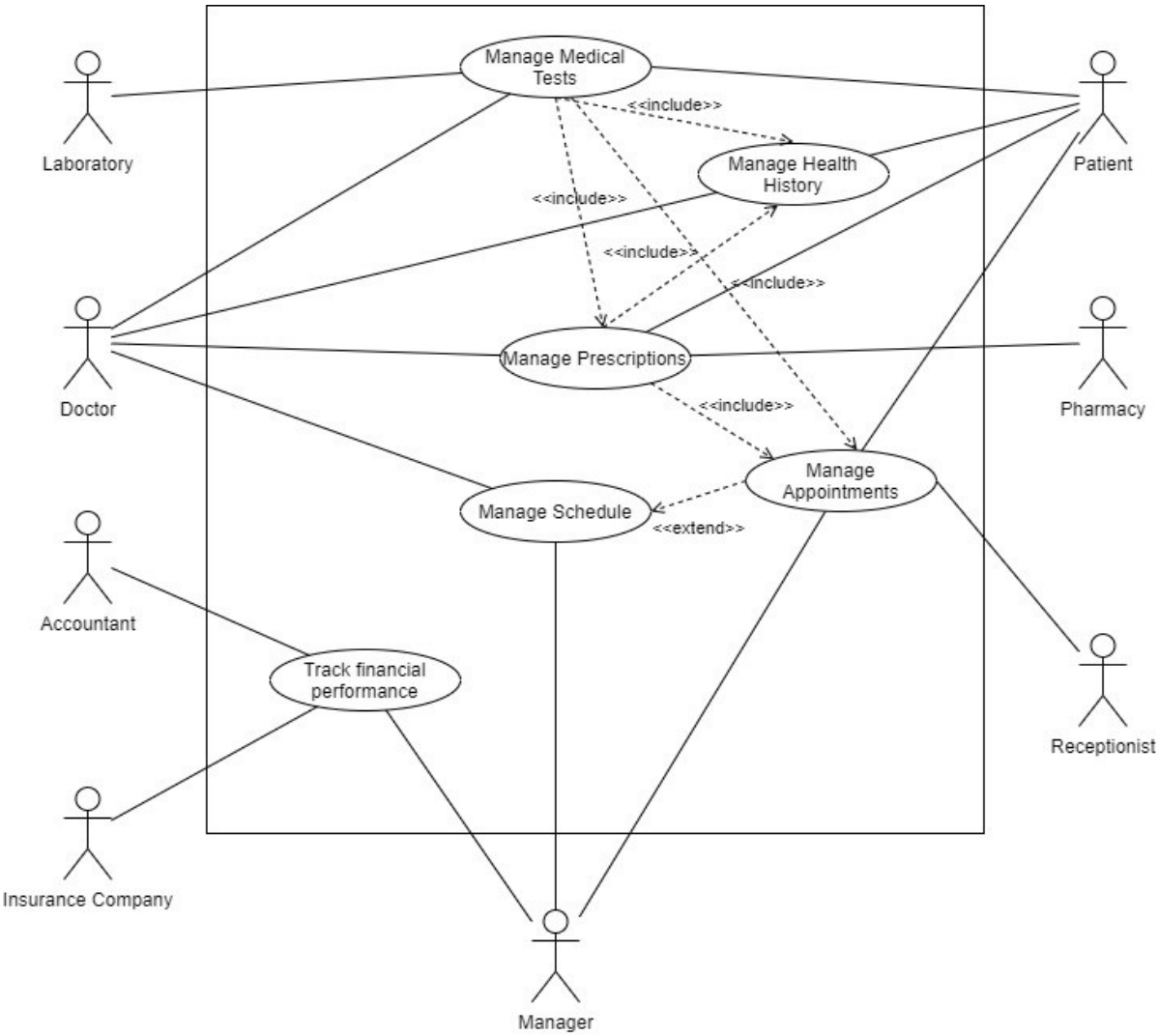

[Patient Declines Testing]
[Patient Accepts Testing]
Order medical testing
Conduct medical test
Share test results
Assess Patient's health and test results
[no concerns]
[some concerns]
Issue a prescription
Send prescription to Pharmacy
Check Patient's condition by phone
[no improvements]
Schedule follow-up visit
[feel better]

2.2 Use Case Diagram

We designed 6 use cases for the medical information system, which cover most of the functions. They are:

- Manage Medical Tests
- Manage Health History
- Manage Prescriptions
- Manage Schedule
- Track Financial Performance
- Manage Appointments

The use cases and the relationships can be seen below in the Use Case Diagram. Detailed Use Case Descriptions start on the next page.



2.3 Use Case Descriptions

Use Case Name: Manage Medical Tests	ID: 1	**Importance level:** High
Primary Actor: Doctor	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: **Doctor** - wants to diagnose patient's medical issues **Patient** - wants medical attention to improve their health **Laboratory** - wants to conduct ordered tests accurately and in a timely manner		
Brief Description: This use case describes how we order, conduct, and evaluate medical tests for a patient.		
Trigger: Doctor determines patient requires medical testing during the appointment **Type:** External		
Relationship: **Association:** Doctor, Patient, Laboratory **Include**: Manage Health History, Manage Appointments, Manage Prescriptions **Extend:** **Generalization:**		
Normal Flow of Events: 1. The doctor meets with the patient during the appointment. 2. The doctor assesses the patient's health. 3. If the doctor recommends medical testing for the patient, The S-1 flow: medical tests sub-flow is performed. 4. If the doctor recommends medication for the patient, Execute Manage Prescriptions use case. 5. If the doctor recommends a follow-up appointment to discuss test results, Execute Manage Appointments use case. 6. The doctor records the outcome of the appointment in the patient's file, Execute Manage Health History use case.		
Sub flows: S-1: Medical Tests 1. The doctor sends the medical test order to the laboratory. 2. The patient goes to the laboratory to have the test conducted. 3. The laboratory completes the medical test. 4. The laboratory sends the test results to the doctor.		
Alternate and Exceptional flows: S-1, 2a1: The patient decides not to have the medical test conducted.		

Use Case Name: Manage Health History	ID: 2	**Importance level:** High
Primary Actor: Doctor	**Use Case Type:** Detail ,Essential	

Stakeholders and Interests:
Doctor - wants to record the patient's disease, surgery, medical examination and medical condition and the date when these diseases occurred or were diagnosed.
Patient - wants to check the patient's own health history.

Brief Description: This use case describes the communication and information interaction between the patient and the doctor, and maintains a high degree of consistency in the transmission of the patient's health history.

Trigger: Patient calls and asks doctor to get their own health history or doctor sends patient an updated health history.
Type: External

Relationship:
Association: Doctor, Patient
Include:
Extend:
Generalization:

Normal Flow of Events:
1. The patient reports to the doctor about health condition.
2. The doctor checks the patient's information.
   If the patient's health condition has no change.
   The S-1: keep the previous health history record subflow is performed.
   If the patient's health condition has changed.
   The S-2: put new health conditions on the health history record subflow is performed.
3. Update the patient health history record.
4. Send the patient health history record.

Subflows:
S - 1 : Keep the previous health history record.
1.The doctor opens the patient's health history database.
2.The doctor uploads the existing health record to the health history database.
3.The doctor sends the results to the patient via database.
S - 2 : Put new health conditions on the health history record.
1.The doctor opens the patient's health history database.
2.The doctor uploads the new health record to the health history database.
3.The doctor sends the results to the patient via database.

Alternate and Exception follow:
S-1, 3a1:The patient decides not to have the health history conducted.
S-2, 3a1:The patient decides not to have the health history conducted.

Use Case Name: Manage Prescriptions	ID: 3	**Importance level:** High
Primary Actor: Doctor	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:
Doctor - wants to proscribe the right medications
Patient - wants to cure his illness
Pharmacy - wants to provide Patient with the prescribed medications

Brief Description: This use case describes how Doctor chooses Patient's treatment and makes sure the treatment is the right one

Trigger: Doctor gets medical test results and/or examines Patient's symptoms
Type: External

Relationship:

Association: Doctor, Patient, Pharmacy
Include:Manage Health History, Manage Appointments
Extend:
Generalization:

Normal Flow of Events:

1. The doctor assesses medical test results and compares them to the symptoms and medical history.
   The S-1: The Doctor draws a conclusion.
2. The doctor chooses medications for the Patient and issues a prescription.
3. Prescription is sent to the Pharmacy.
   The S-2: Purchase medications subflow is performed.
4. The doctor calls the Patient to check his condition.
5. If the Patient isn't getting better, the Doctor schedules follow-up appointment.
   Execute Manage Appointments use case.

Sub follows:

The S-1: The Doctor draws a conclusion:

1. The Doctor checks medical test results and compares them to standards and/or previous results and symptoms.
2. The Doctor checks medical history for possible insights.
3. The Doctor advises proper medications.
4. The Doctor updates Health History.

The S-2: Purchase medications:

1. The doctor sends a prescription to the Pharmacy.
2. The Pharmacy checks availability of the medications.
3. The Pharmacy confirms to the Patient availability of the medications.
4. The Patient collects the medications from the Pharmacy.

Alternate and Exception follow:

S-1, 3a: If there's no reason to worry about, the Doctor finishes the visit.
S-2, 2a: The medication isn't available. The Pharmacy informs the Patient.

Use Case Name: Manage Schedule	ID: 4	**Importance level:** High
Primary Actor: Manager	**Use Case Type:** Detailed, Essential	

Stakeholders and Interests:

Manager - wants to track financial performance and/or manage appointments
Doctor - wants to make sure patient's medical necessities are addressed

Brief Description: This use case describe how we track financial performance as well as manage appointments

Trigger: Manager requests doctor's availability and quarterly financial report

Type: External

Relationship:

Association: Manager
Include:
Extend: Manage Appointments
Generalization:

Normal Flow of Events:

1. Manager contacts receptionists regarding scheduling
2. Receptionist provides manager with booked appointments
3. If doctor's schedule has a cancelled appointment:
    1. Rebook later appointment to earlier appointments
4. Manager contacts accountant regarding successful transactions
5. Accountant provides manager weekly transitions report
6. Manager provides doctor with "bottom line" progress assessment

Sub follows:

S-1: Manage Schedules
1. The Manager sends doctors' availability to receptionists quarterly.
2. Receptionists update doctors' weekly schedule accordingly.

Alternate and Exception follow:

3.1 If doctor's appointment cancelled Manage Appointments is activated and later appointments are upgraded to earlier schedule

Use Case Name: Track Financial Performance	ID: 5	**Importance level:** High
Primary Actor: Manager	**Use Case Type:** Detail ,Essential	

Stakeholders and Interests:
Manager - wants to see information about appointed patients such as an itemized list of the amount owed for the services provided, including any tests and prescriptions administered.
Insurance company – covers payments for the insured patients according to the policy.
Accountant - Controls patient's billing and payments.

Brief Description: This use case describes how manager tracks patient's payments in the company.

Trigger: Manager calls accountant asks for check appointed patients 's balance.
Type: External

Relationships:
Association: Accountant, Insurance Company, Manager
Include:
Extend:
Generalization:

Normal Flow of Events:

1. The manager contacts accountant to request to check the appointed patient's balance.
2. The accountant verifies manager's permission number.
3. The accountant calculates all the amount of financial bill.
   If the patient is without insurance coverage.
   The S-1 :The accountant calculates the amount directly without insurance coverage subflow is performed.
   If the patient is with insurance coverage.
   The S - 2 :The accountant will send a request to the insurance company subflow is performed.
4. The insurance company assists the accountant, checks the patient with insurance coverage.
   If insurance company rejects insurance indemnity
   The S -3 :rejection information subflow is performed.
   If insurance company agrees to the insurance claim.
   The S- 4: calculation of amount information subflow is performed.
5. Accountant sends all the financial bills to manager.
   The S -5 : calculate the total financial bill subflow is performed.

Subfollows:

S-1: The accountant calculates the amount directly without coverage.
S-2: The accountant will send a request to insurance company.
1. The insurance company verifies the insurance number.
2. The insurance company sends feedback information if the insurance number is present.
S-3: Insurance company sends rejected information to accountant.
1. The insurance company looks up the claim number in the insurance company's database.
2. The insurance company verifies the claim conditions via insurance company's database.
3. The insurance company sends rejection information to the accountant.
S-4 : The insurance company calculates the amount of insurance indemnity to the accountant.
1. The insurance company looks up the claim number in the insurance company's database.
2. The insurance company verifies the claim condition via insurance company's database.
3. The insurance company sends the claim amount to accountant.
S-5: The accountant packs up two kinds of financial bills.
1. The accountant uploads two kinds of patient 's bills(with insurance coverage or without insurance coverage) to 'Account Check Bill Balance Database'.
2. The accountant response billing results to manager.

Alternate and Exception follow:

1a-1: The manager failed to send a request to accountant.

Use Case Name: Manage Appointments	ID: 6	**Importance level:** Low
Primary Actor: Patient	**Use Case Type:** Detail, Essential	

Stakeholders and Interests:

Receptionist: make sure to prepare the patient's information for the doctors
Patient: wants to make, change or cancel an appointment
Manager: interested in the maximum number of appointments for the doctor.

Brief Description: This is a use case description about how to make an appointment as well as changing or canceling an appointment.

Trigger: When the patient contacts the office about new appointment or cancel appointments or change appointment

Type: External

Relationship:

Association: Patient, Receptionist, Manager
Include:
Extend:
Generalization:

Normal Flow of Events:

1. Patient contacts the receptionist regarding a medical appointment
2. The patient provides the staff with the name and address
    1. If the patient information is not found
        1. The new patient subflow is performed
3. If the patient information has changed
    1. Execute the update information subflow
4. The receptionist asks the patient if he wants to make an appointment or change an appointment or cancel the previous appointment.
    1. If the patient wants to make a new appointment
        1. The new appointment subflow is performed
    2. If the patient wants to change an existing appointment
        1. The change appointment subflow is performed
    3. If the patient wants to cancel an existing appointment
        1. The change appointment subflow is performed
5. The receptionist provides the results of the transaction to the patient.

Sub follows:

S1, New patient

1. The Receptionist asks more basic information about the Patient
2. The Receptionist creates a new patient's account
3. The Receptionist asks patient about their possible appointment times
4. The Receptionist matches patient's desired times with available dates and times, then schedules the new appointment

S2 Update information

1. The Receptionist asks about the new information
2. The Receptionist changes the patient's information

S3, New appointment

1. The Receptionist asks patient about their possible appointment times
2. The Receptionist matches patient's desired times with available dates and times, then schedules the new appointment

S4, Cancel Appointment

1. The Receptionist asks the Patient for the old appointment time.
2. The Receptionist finds the current appointment in the appointment file and cancels it.

S5, Change Appointment

1. The Receptionist performs the S-4: cancel appointment subflow.
2. The Receptionist performs the S-3: new appointment subflow.

Alternate and Exception follow:

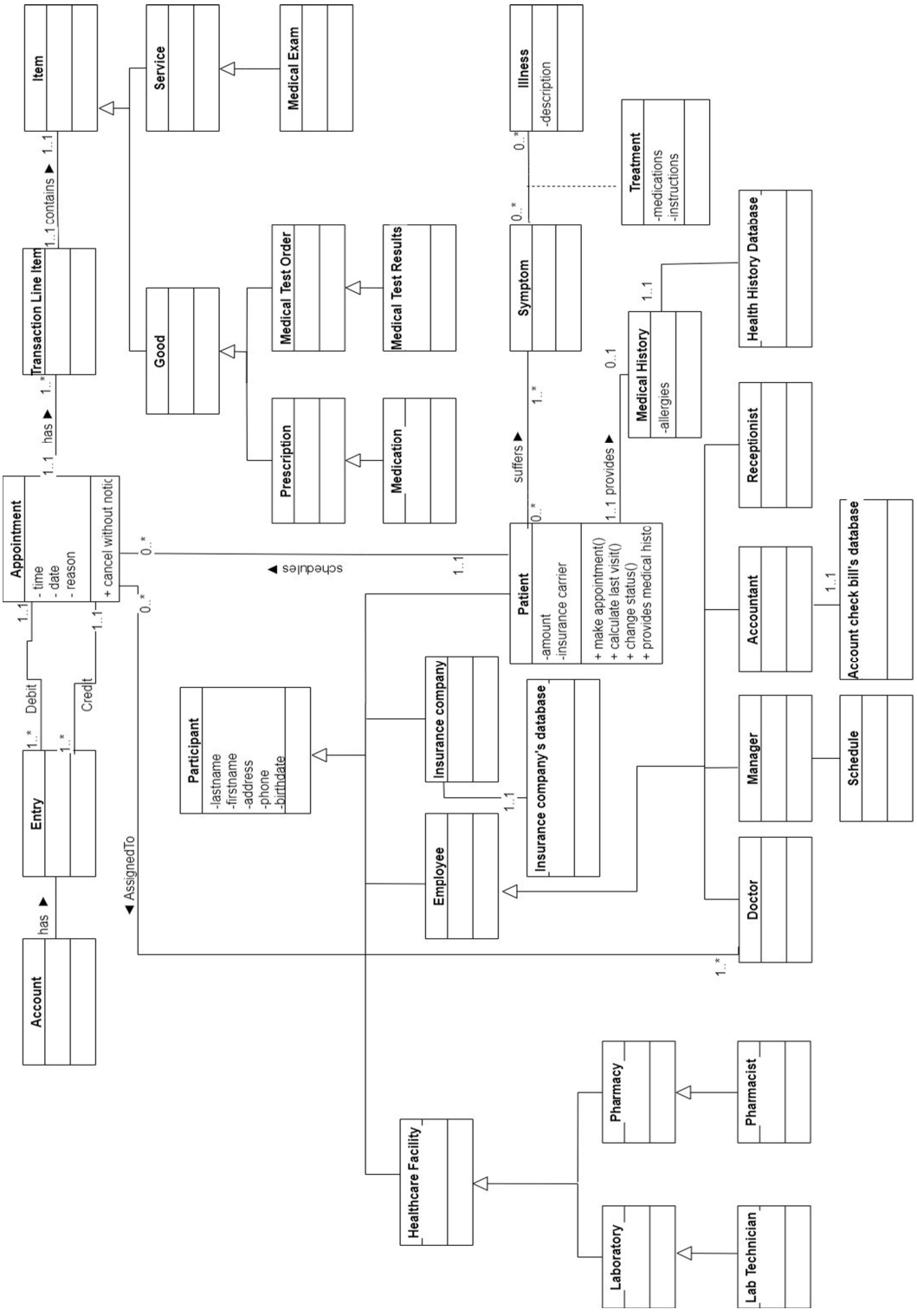
S1.2.1 the new patient basic information is incomplete

S3.2.1 The Receptionist proposes some alternative appointment times based on what is available in the appointment schedule

S3.2.2 The patient chooses one of the proposed times or decides not to make an appointment

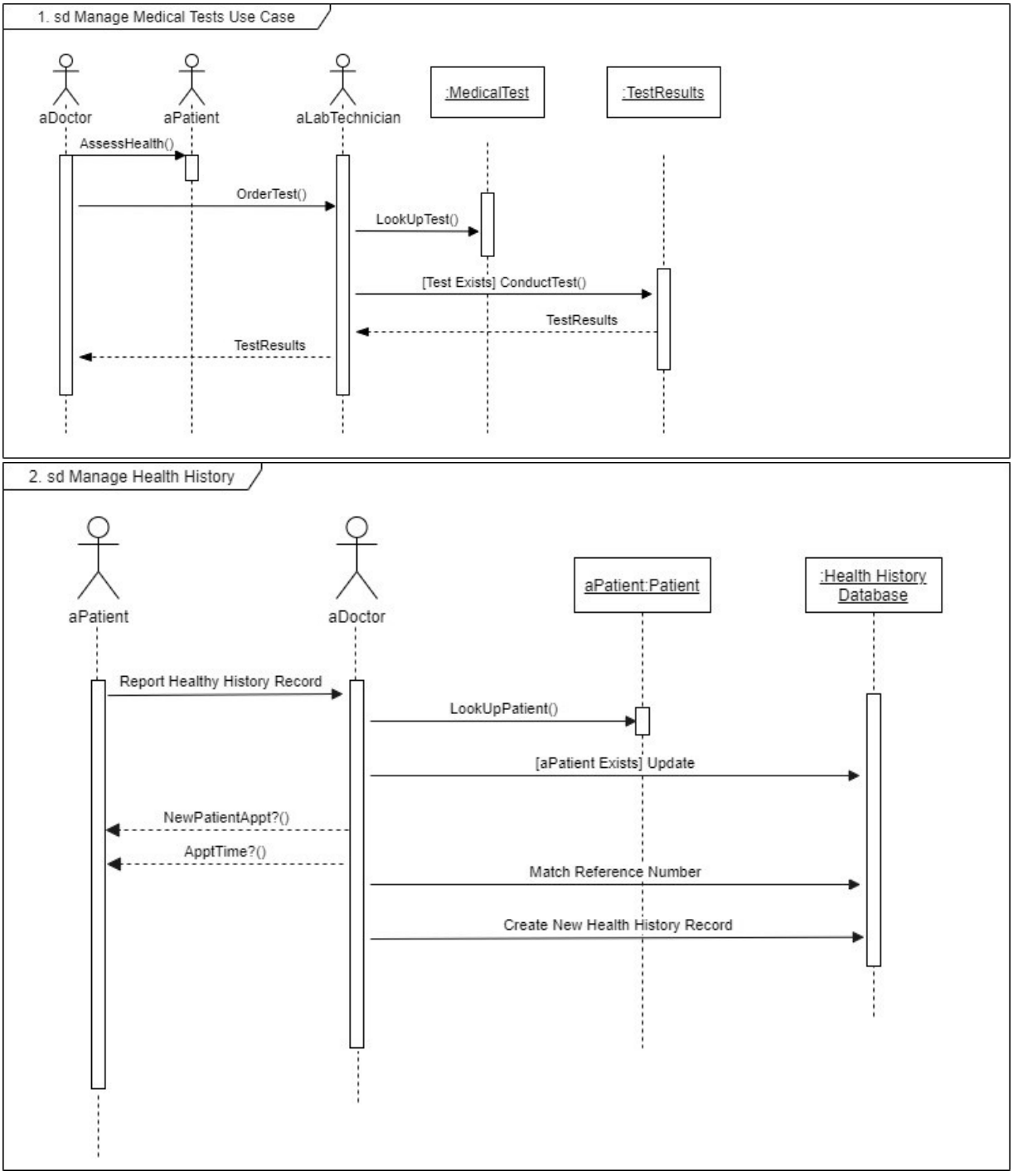
2.4 Initial Class Diagram

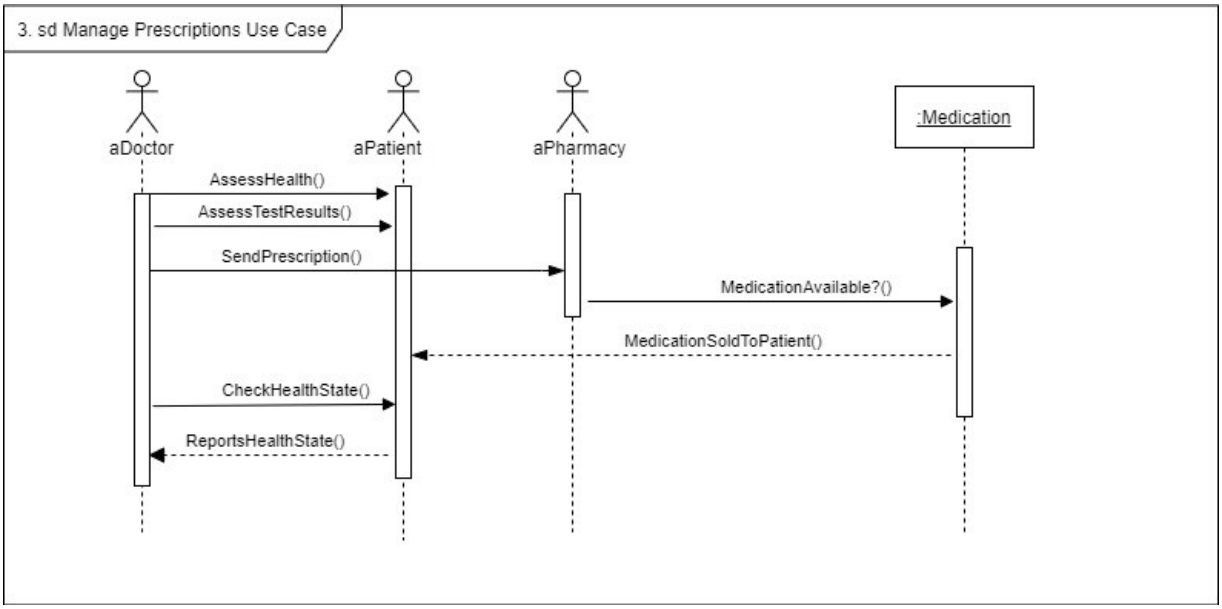
The initial class diagram is shown below.

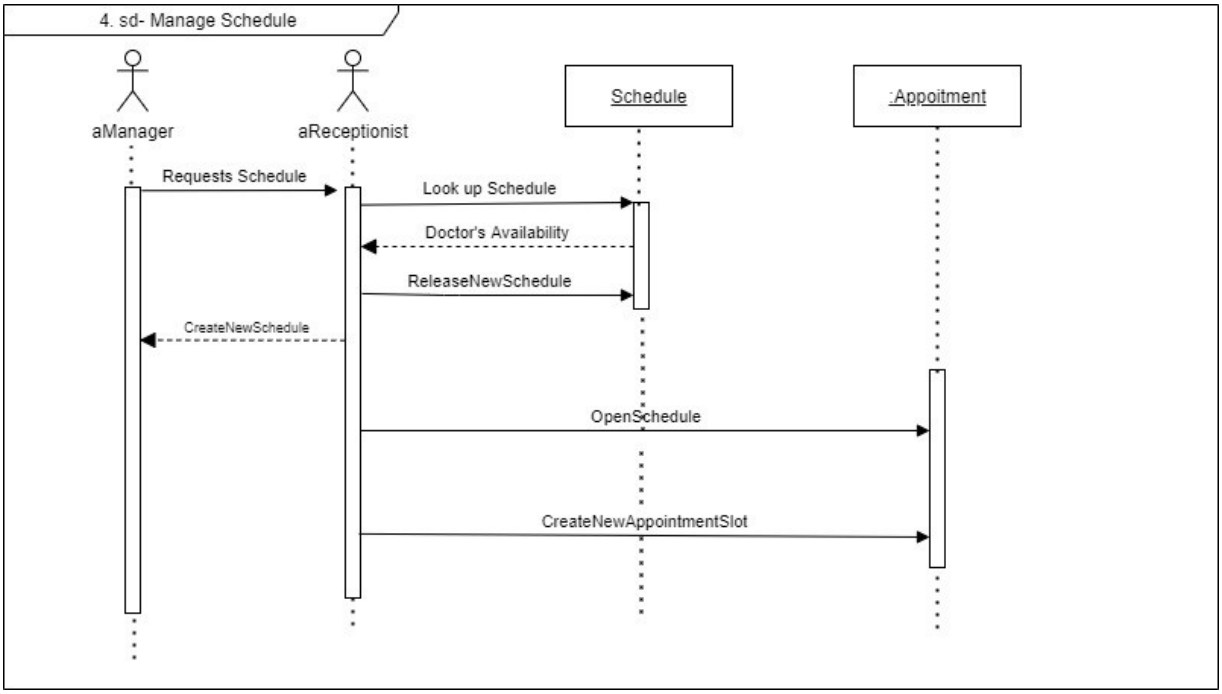


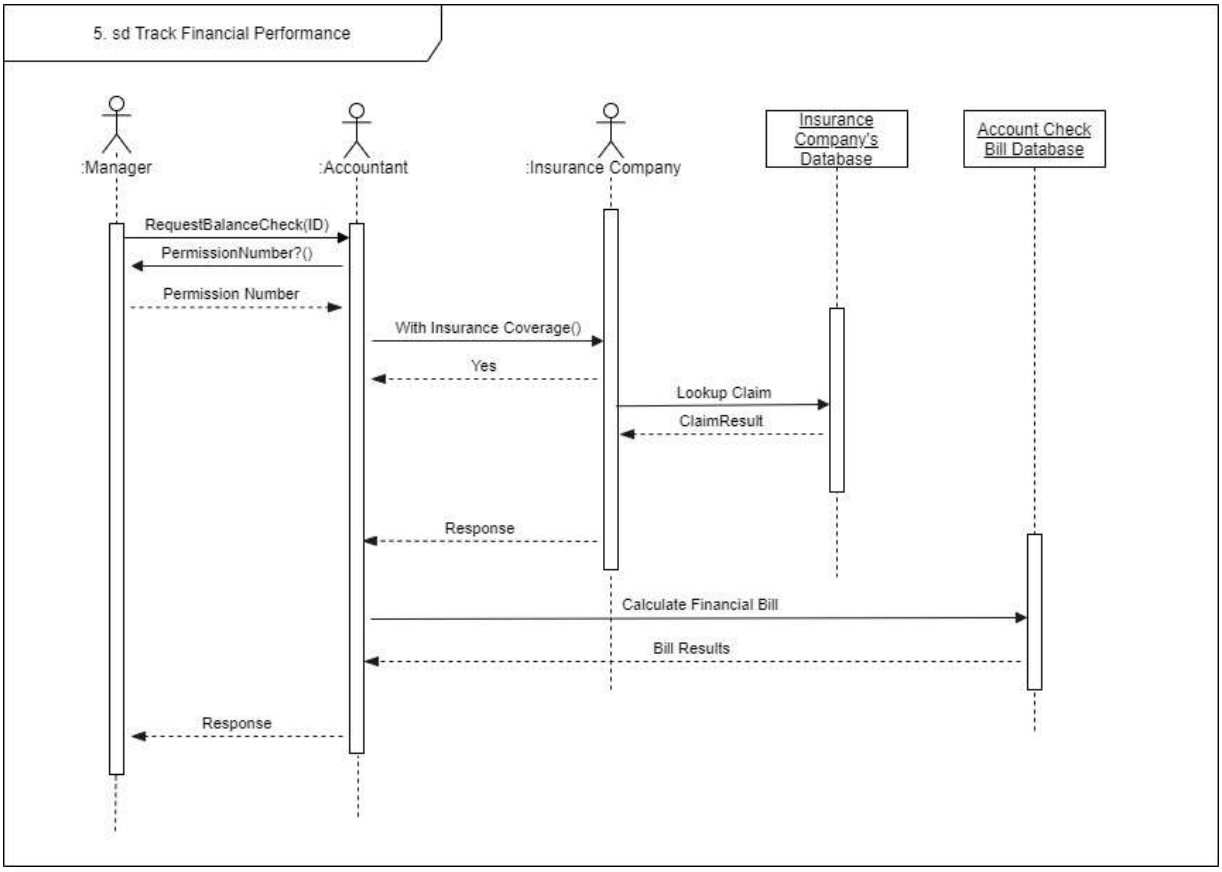
2.5 Sequence Diagrams

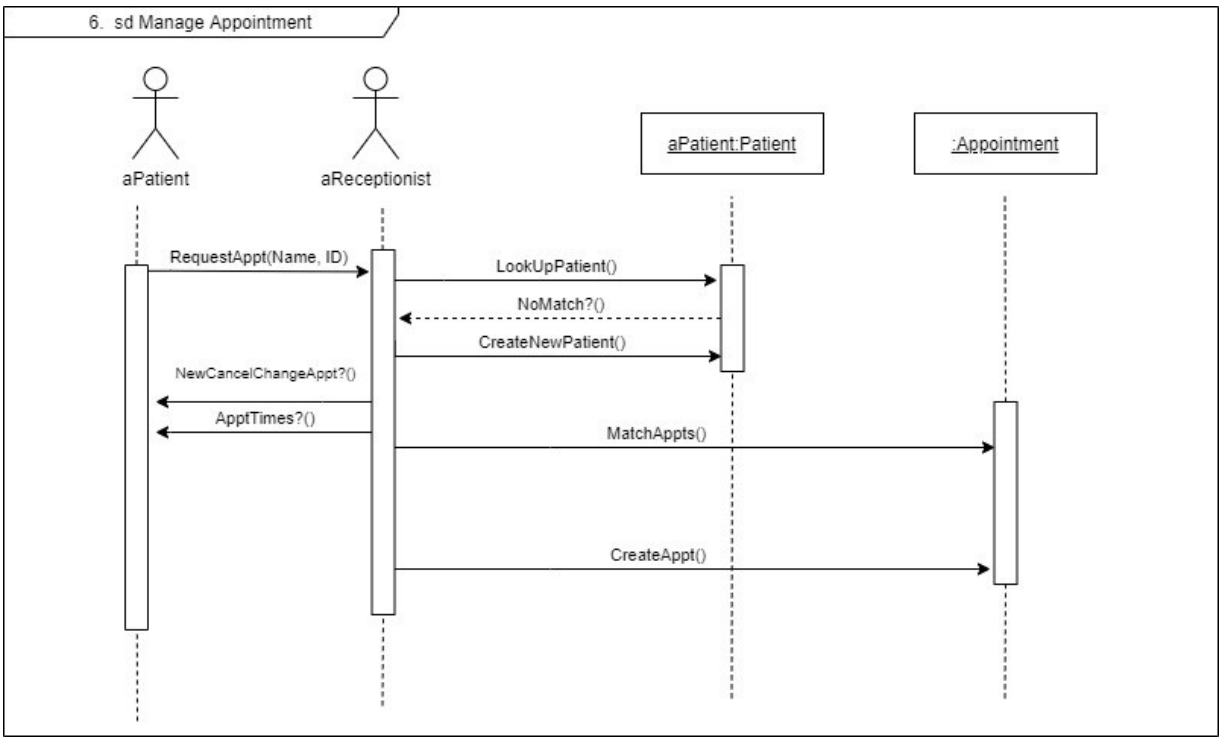
The sequence diagrams show the objects that participate in the use cases and the messages that are sent between them over time. Each diagram corresponds to one use case.




3. sd Manage Prescriptions Use Case
aDoctor
aPatient
aPharmacy
:Medication
AssessHealth()
AssessTestResults()
SendPrescription()
MedicationAvailable?()
MedicationSoldToPatient()
CheckHealthState()
ReportsHealthState()


4. sd- Manage Schedule
aManager
aReceptionist
Schedule
:Appoitment
Requests Schedule
Look up Schedule
Doctor's Availability
ReleaseNewSchedule
CreateNewSchedule
OpenSchedule
CreateNewAppointmentSlot

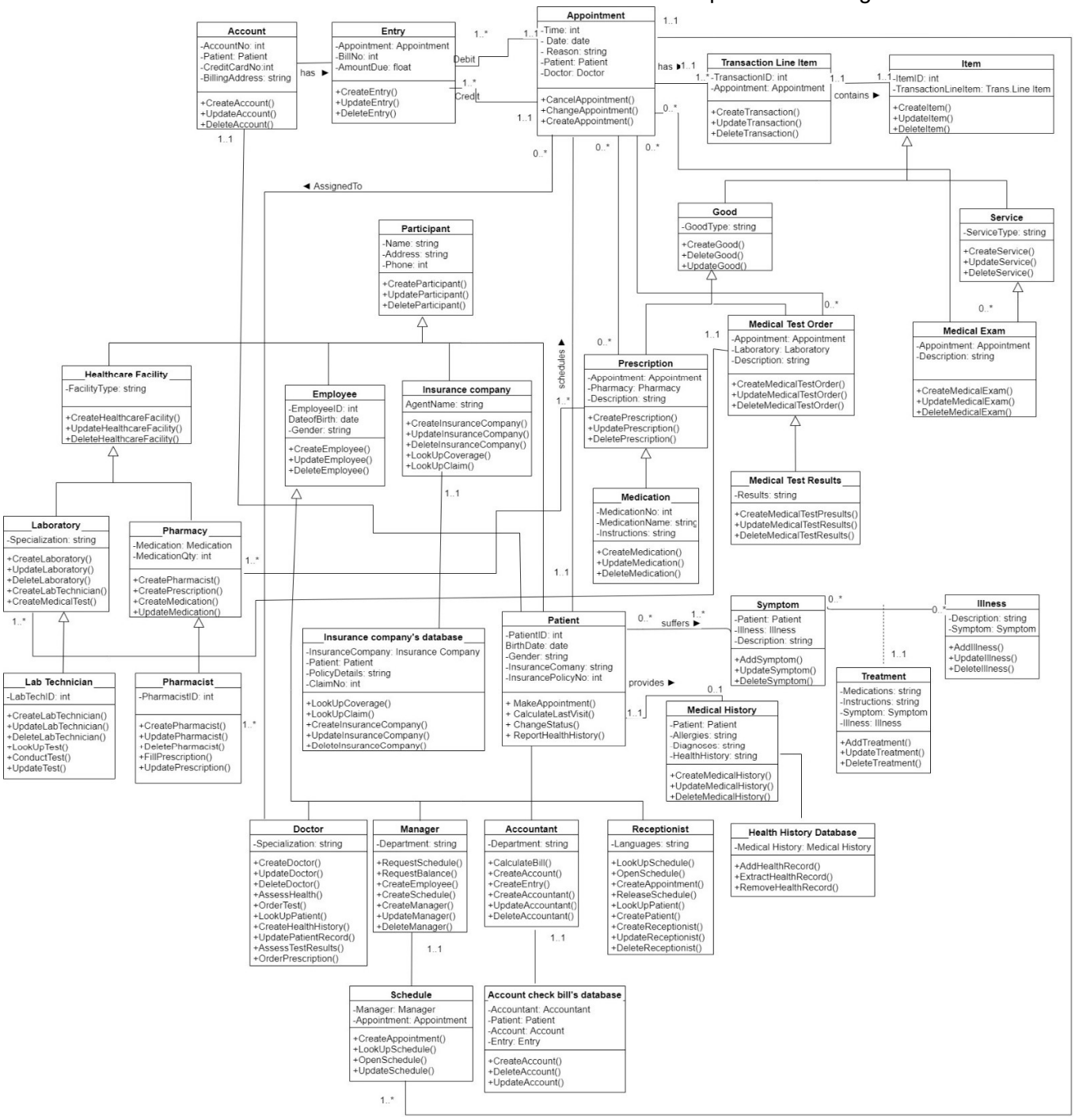

5. sd Track Financial Performance
:Manager
:Accountant
:Insurance Company
Insurance Company's Database
Account Check Bill Database
RequestBalanceCheck(ID)
PermissionNumber?()
Permission Number
With Insurance Coverage()
Yes
Lookup Claim
ClaimResult
Response
Calculate Financial Bill
Bill Results
Response


6. sd Manage Appointment
aPatient
aReceptionist
aPatient:Patient
:Appointment
RequestAppt(Name, ID)
LookUpPatient()
NoMatch?()
CreateNewPatient()
NewCancelChangeAppt?()
ApptTimes?()
MatchAppts()
CreateAppt()

3. Design

3.1 Final class diagram

The final class diagram is shown below. It is slightly different than the initial one. We realized that some of the references between classes were incorrect and some relationships were missing.



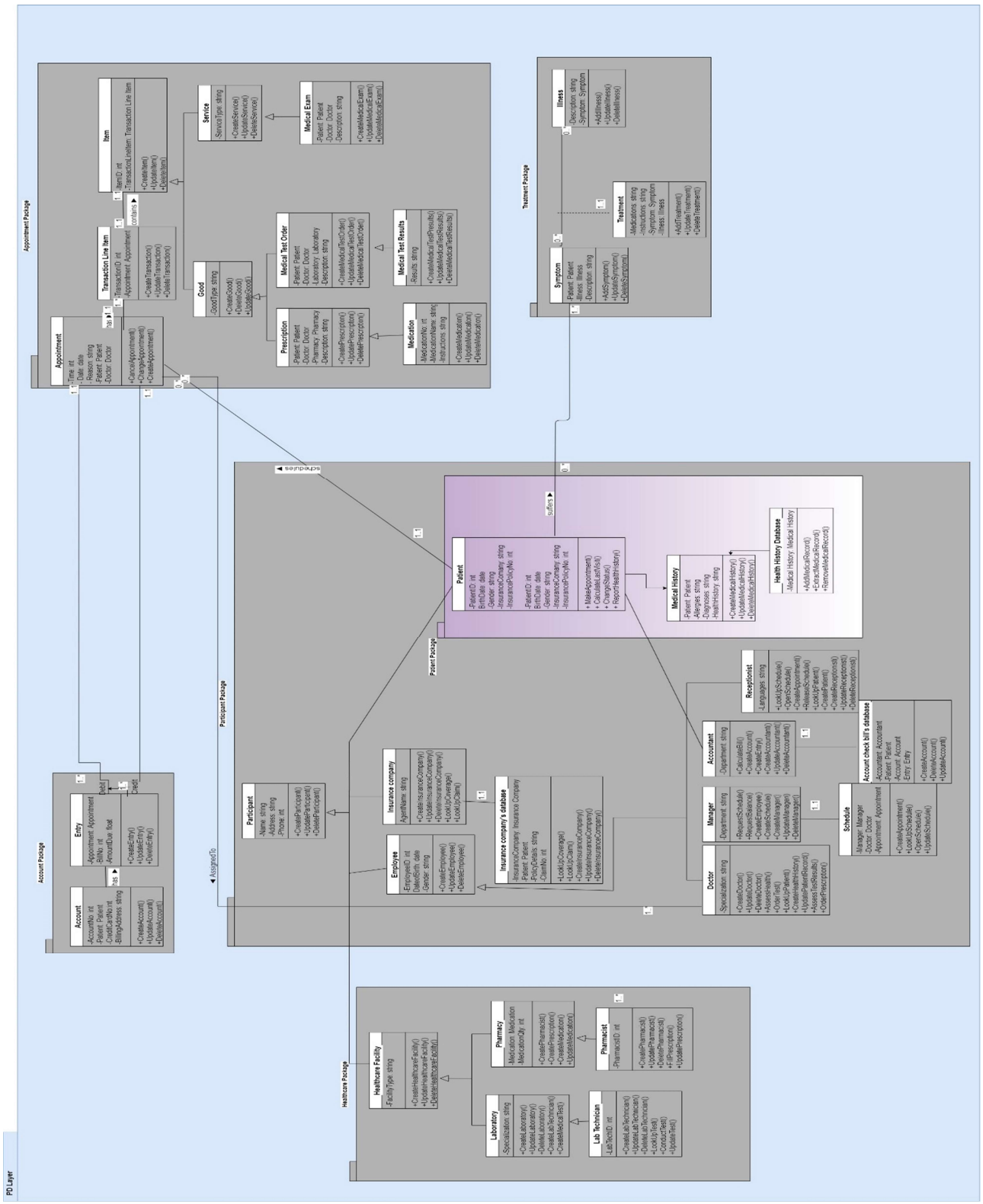
3.2 Package diagram

The package diagram is shown below.
Account class and Entry class are packaged together as Account Package.

Appointment class, Transaction Line item class, Item class, Good class, Prescription class, Medical Test Order class, Medication class, Medical class, Service class and Medical Exam class are packaged together as Appointment Package.

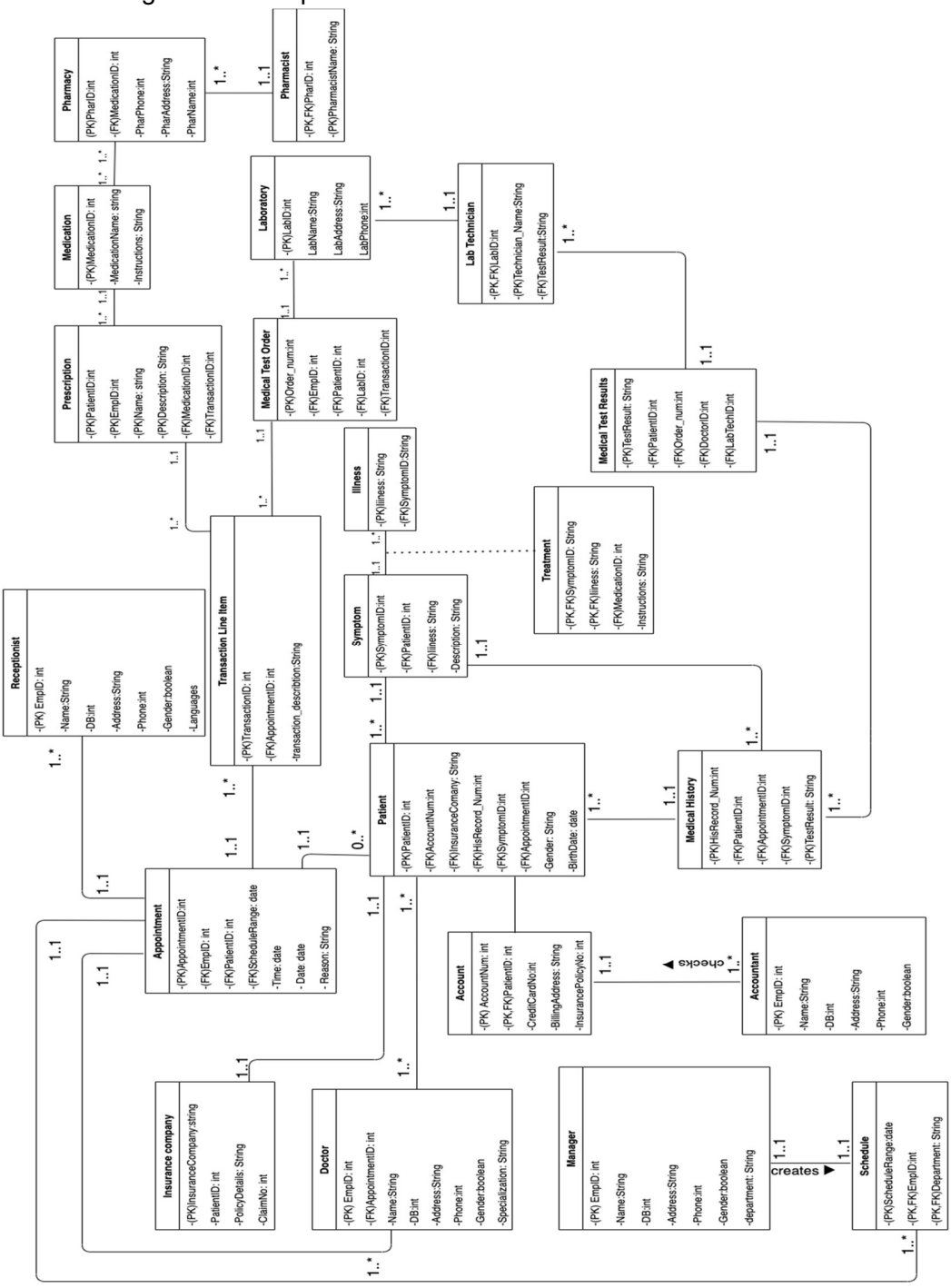
Healthcare Facility class, Laboratory class, Pharmacy class, Lab Technician class and Pharmacist class are packaged together as Healthcare Package.

Participant Package, Patient Package and Treatment Package.


PD Layer
Appointment Package
Appointment
-Time int
-Date date
-Reason string
-Patient Patient
-Doctor Doctor
+CancelAppointment()
+ChangeAppointment()
+CreateAppointment()
Transaction Line Item
-TransactionID int
-Appointment Appointment
+CreateTransaction()
+UpdateTransaction()
+DeleteTransaction()
Item
-ItemID int
-TransactionLineItem Transaction Line Item
+CreateItem()
+UpdateItem()
+DeleteItem()
contains
Service
-ServiceType string
+CreateService()
+UpdateService()
+DeleteService()
Medical Exam
-Patient Patient
-Doctor Doctor
-Description string
+CreateMedicalExam()
+UpdateMedicalExam()
+DeleteMedicalExam()
Good
-GoodType string
+CreateGood()
+DeleteGood()
+UpdateGood()
Medical Test Order
-Patient Patient
-Doctor Doctor
-Laboratory Laboratory
-Description string
+CreateMedicalTestOrder()
+UpdateMedicalTestOrder()
+DeleteMedicalTestOrder()
Medical Test Results
-Results string
+CreateMedicalTestResults()
+UpdateMedicalTestResults()
+DeleteMedicalTestResults()
Prescription
-Patient Patient
-Doctor Doctor
-Pharmacy Pharmacy
-Description string
+CreatePrescription()
+UpdatePrescription()
+DeletePrescription()
Medication
-MedicationNo int
-MedicationName string
-Instructions string
+CreateMedication()
+UpdateMedication()
+DeleteMedication()
Treatment Package
Illness
-Description string
-Symptom Symptom
+AddIllness()
+UpdateIllness()
+DeleteIllness()
Symptom
-Patient Patient
-Illness Illness
-Description string
+AddSymptom()
+UpdateSymptom()
+DeleteSymptom()
Treatment
-Medications string
-Instruction string
-Symptom Symptom
-Illness Illness
+AddTreatment()
+UpdateTreatment()
+DeleteTreatment()
Account Package
Account
-AccountNo int
-Patient Patient
-Current Balance int
-BillingAddress string
+CreateAccount()
+UpdateAccount()
+DeleteAccount()
has
Entry
-Appointment Appointment
-BillNo int
-AmountDue float
+CreateEntry()
+UpdateEntry()
+DeleteEntry()
Debit
Credit
Participant Package
Participant
-Name string
-Address string
-Phone int
+CreateParticipant()
+UpdateParticipant()
+DeleteParticipant()
Insurance company
-AgentName string
+CreateInsuranceCompany()
+UpdateInsuranceCompany()
+DeleteInsuranceCompany()
+LookUpCoverage()
+LookUpClaim()
Employee
-EmployeeID int
-DateofBirth date
-Gender string
+CreateEmployee()
+UpdateEmployee()
+DeleteEmployee()
Insurance company's database
-InsuranceCompany Insurance Company
-PatientDetails string
-PolicyDetails string
-ClaimNo int
+LookUpCoverage()
+LookUpClaim()
+CreateInsuranceCompany()
+UpdateInsuranceCompany()
+DeleteInsuranceCompany()
Patient Package
Patient
-PatientID int
-BirthDate date
-Gender string
-InsuranceCompany string
-InsurancePolicyNo int
+MakeAppointment()
+CalculateLastVisit()
+ChangeStatus()
+ReportHealthHistory()
Medical History
-Patient Patient
-Allergies string
-Diseases string
-HealthHistory string
+CreateMedicalHistory()
+UpdateMedicalHistory()
+DeleteMedicalHistory()
Health History Database
-Medical History Medical History
+AddMedicalRecord()
+DrawMedicalRecord()
+RemoveMedicalRecord()
Receptionist
-Languages string
+LookUpSchedule()
+OpenSchedule()
+CreateAppointment()
+ReleaseSchedule()
+LookUpPatient()
+CreatePatient()
+UpdateReceptionist()
+DeleteReceptionist()
Accountant
-Department string
+CalculateBill()
+PostBill()
+CreateEntry()
+CreateAccount()
+UpdateAccount()
+DeleteAccount()
Doctor
-Specialization string
+CreateDoctor()
+UpdateDoctor()
+DeleteDoctor()
+AssessHealth()
+OrderTest()
+LookUpPatient()
+CreateHealthHistory()
+UpdateMedicalRecord()
+AssessTestResults()
+OrderPrescription()
Manager
-Department string
+RequestSchedule()
+ReleaseSchedule()
+CreateEmployee()
+CreateSchedule()
+CreateManager()
+UpdateManager()
+DeleteManager()
Schedule
-Manager Manager
-Doctor Doctor
-Appointment Appointment
+CreateAppointment()
+LookUpSchedule()
+OpenSchedule()
+UpdateSchedule()
Account check bill's database
-Accountant Accountant
-Account Account
-Account Record
Entry Entry
+CreateAccount()
+DeleteAccount()
+UpdateAccount()
Healthcare Package
Healthcare Facility
-FacilityType string
+CreateHealthcareFacility()
+UpdateHealthcareFacility()
+DeleteHealthcareFacility()
Laboratory
-Specialization string
+CreateLaboratory()
+UpdateLaboratory()
+DeleteLaboratory()
+CreateLabTechnician()
+CreateMedicalTest()
Pharmacy
-Medication Medication
-MedicationQty int
+CreatePharmacist()
+CreatePrescription()
+CreateMedication()
+UpdateMedication()
Lab Technician
-LabTechID int
+CreateLabTechnician()
+UpdateLabTechnician()
+DeleteLabTechnician()
+AddLabTest()
+ConductTest()
+UpdateTest()
Pharmacist
-PharmacistID int
+CreatePharmacist()
+UpdatePharmacist()
+DeletePharmacist()
+FillPrescription()
+UpdatePrescription()
schedules
Suffers
AssignedTo

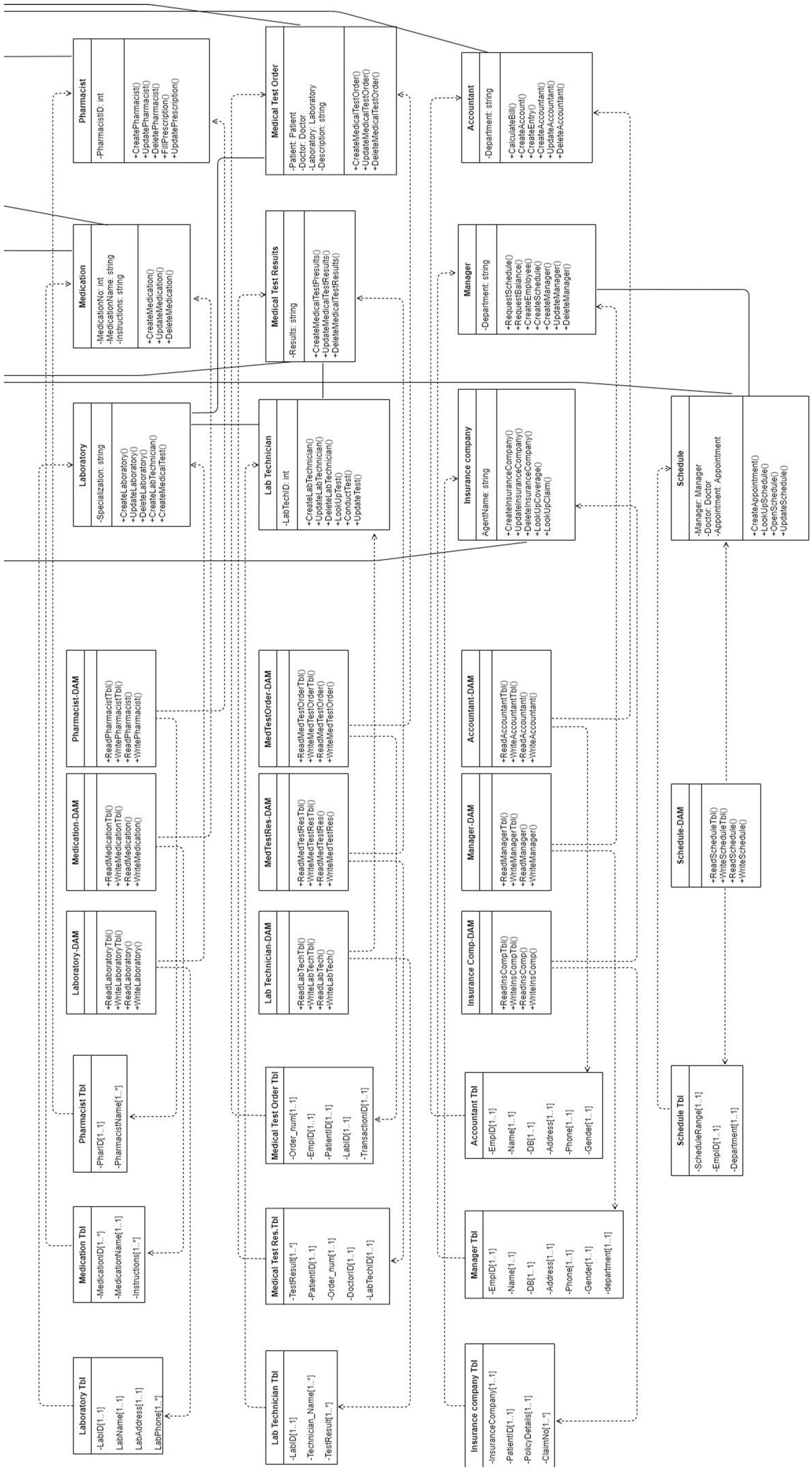
3.3 Database design

The Database design shows the association among the instances with their databases. The new change is we cut off the abstract classes from the final class diagram and maintain the concrete classes at our DB design. The example below illustrates the connections:



3.4 Data Access and Manipulation Design

The Data Access and Manipulation design illustrates the interaction among the classes with their databases correspondents. The DAM are utilized for connection purposes only by the classes themselves. The example below illustrates the connections:

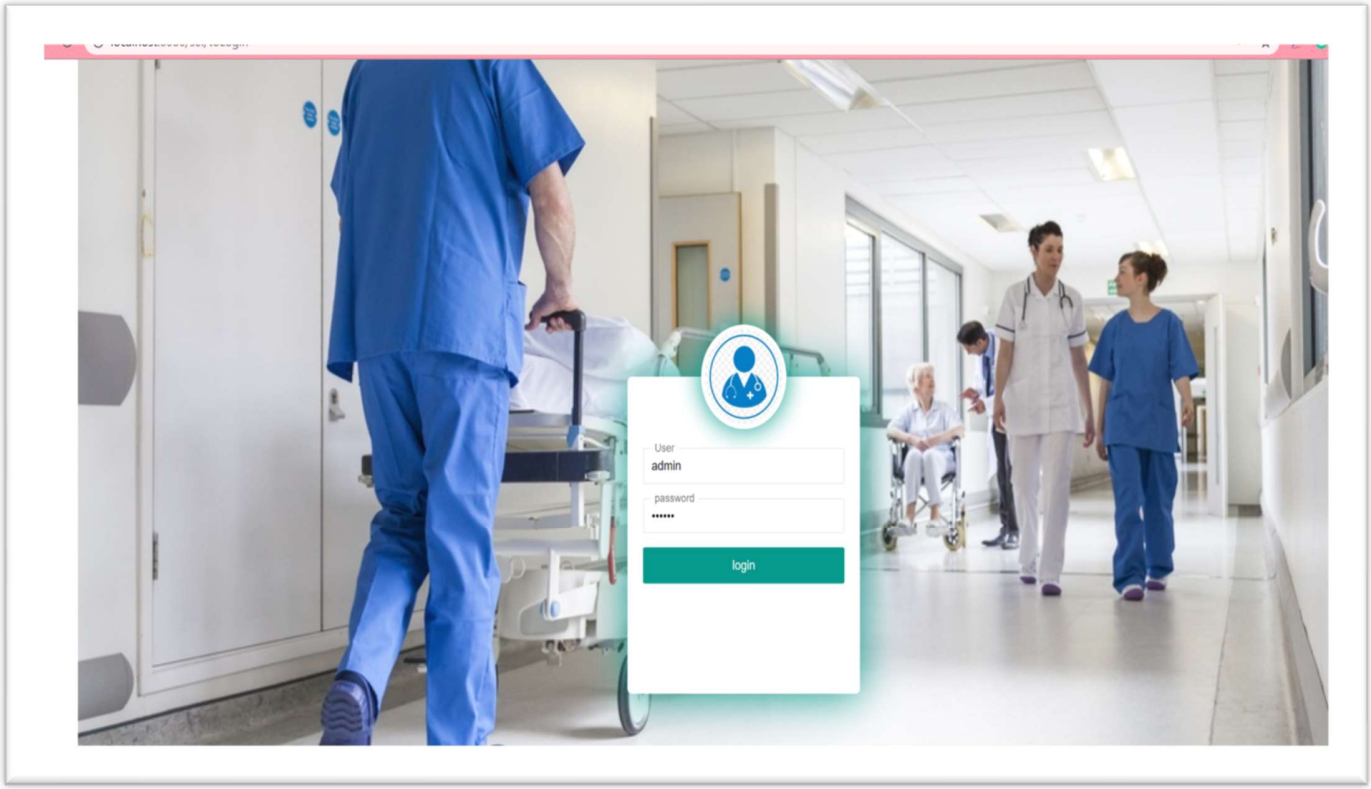

Pharmacist
-PharmacistID: int
+CreatePharmacist()
+UpdatePharmacist()
+DeletePharmacist()
+FillPrescription()
+UpdatePrescription()
Medication
-MedicationNo: int
-MedicationName: string
-Instructions: string
+CreateMedication()
+UpdateMedication()
+DeleteMedication()
Laboratory
-Specialization: string
+CreateLaboratory()
+UpdateLaboratory()
+DeleteLaboratory()
+CreateLabTechnician()
+CreateMedicalTest()
Medical Test Order
-Patient: Patient
-Doctor: Doctor
-Laboratory: Laboratory
-Description: string
+CreateMedicalTestOrder()
+UpdateMedicalTestOrder()
+DeleteMedicalTestOrder()
Medical Test Results
-Results: string
+CreateMedicalTestPresults()
+UpdateMedicalTestResults()
+DeleteMedicalTestResults()
Lab Technician
-LabTechID: int
+CreateLabTechnician()
+UpdateLabTechnician()
+DeleteLabTechnician()
+LookUpTest()
+ConductTest()
+UpdateTest()
Accountant
-Department: string
+CalculateBill()
+CreateAccount()
+CreateEntry()
+CreateAccountant()
+UpdateAccountant()
+DeleteAccountant()
Manager
-Department: string
+RequestSchedule()
+RequestBalance()
+CreateEmployee()
+CreateSchedule()
+CreateManager()
+UpdateManager()
+DeleteManager()
Insurance company
-AgentName: string
+CreateInsuranceCompany()
+UpdateInsuranceCompany()
+DeleteInsuranceCompany()
+LookUpCoverage()
+LookUpClaim()
Schedule
-Manager: Manager
-Doctor: Doctor
-Appointment: Appointment
+CreateAppointment()
+LookUpSchedule()
+OpenSchedule()
+UpdateSchedule()
Pharmacist-DAM
+ReadPharmacistTbl()
+WritePharmacistTbl()
+ReadPharmacist()
+WritePharmacist()
Medication-DAM
+ReadMedicationTbl()
+WriteMedicationTbl()
+ReadMedication()
+WriteMedication()
Laboratory-DAM
+ReadLaboratoryTbl()
+WriteLaboratoryTbl()
+ReadLaboratory()
+WriteLaboratory()
MedTestOrder-DAM
+ReadMedTestOrderTbl()
+WriteMedTestOrderTbl()
+ReadMedTestOrder()
+WriteMedTestOrder()
MedTestRes-DAM
+ReadMedTestResTbl()
+WriteMedTestResTbl()
+ReadMedTestRes()
+WriteMedTestRes()
Lab Technician-DAM
+ReadLabTechTbl()
+WriteLabTechTbl()
+ReadLabTech()
+WriteLabTech()
Accountant-DAM
+ReadAccountantTbl()
+WriteAccountantTbl()
+ReadAccountant()
+WriteAccountant()
Manager-DAM
+ReadManagerTbl()
+WriteManagerTbl()
+ReadManager()
+WriteManager()
Insurance Comp-DAM
+ReadInsCompTbl()
+WriteInsCompTbl()
+ReadInsComp()
+WriteInsComp()
Schedule-DAM
+ReadScheduleTbl()
+WriteScheduleTbl()
+ReadSchedule()
+WriteSchedule()
Pharmacist Tbl
-PharID[1..1]
-PharmacistName[1..*]
Medication Tbl
-MedicationID[1..*]
-MedicationName[1..1]
-Instructions[1..*]
Laboratory Tbl
-LabID[1..1]
LabName[1..1]
LabAddress[1..1]
LabPhone[1..*]
Medical Test Order Tbl
-Order_num[1..1]
-EmpID[1..1]
-PatientID[1..1]
-LabID[1..1]
-TransactionID[1..1]
Medical Test Res.Tbl
-TestResult[1..*]
-PatientID[1..1]
-Order_num[1..1]
-DoctorID[1..1]
-LabTechID[1..1]
Lab Technician Tbl
-LabID[1..1]
-Technician_Name[1..*]
-TestResult[1..*]
Accountant Tbl
-EmpID[1..1]
-Name[1..1]
-DB[1..1]
-Address[1..1]
-Phone[1..1]
-Gender[1..1]
Manager Tbl
-EmpID[1..1]
-Name[1..1]
-DB[1..1]
-Address[1..1]
-Phone[1..1]
-Gender[1..1]
-department[1..1]
Insurance company Tbl
-InsuranceCompany[1..1]
-PatientID[1..1]
-PolicyDetails[1..1]
-ClaimNo[1..*]
Schedule Tbl
-ScheduleRange[1..1]
-EmpID[1..1]
-Department[1..1]

4. Implementation

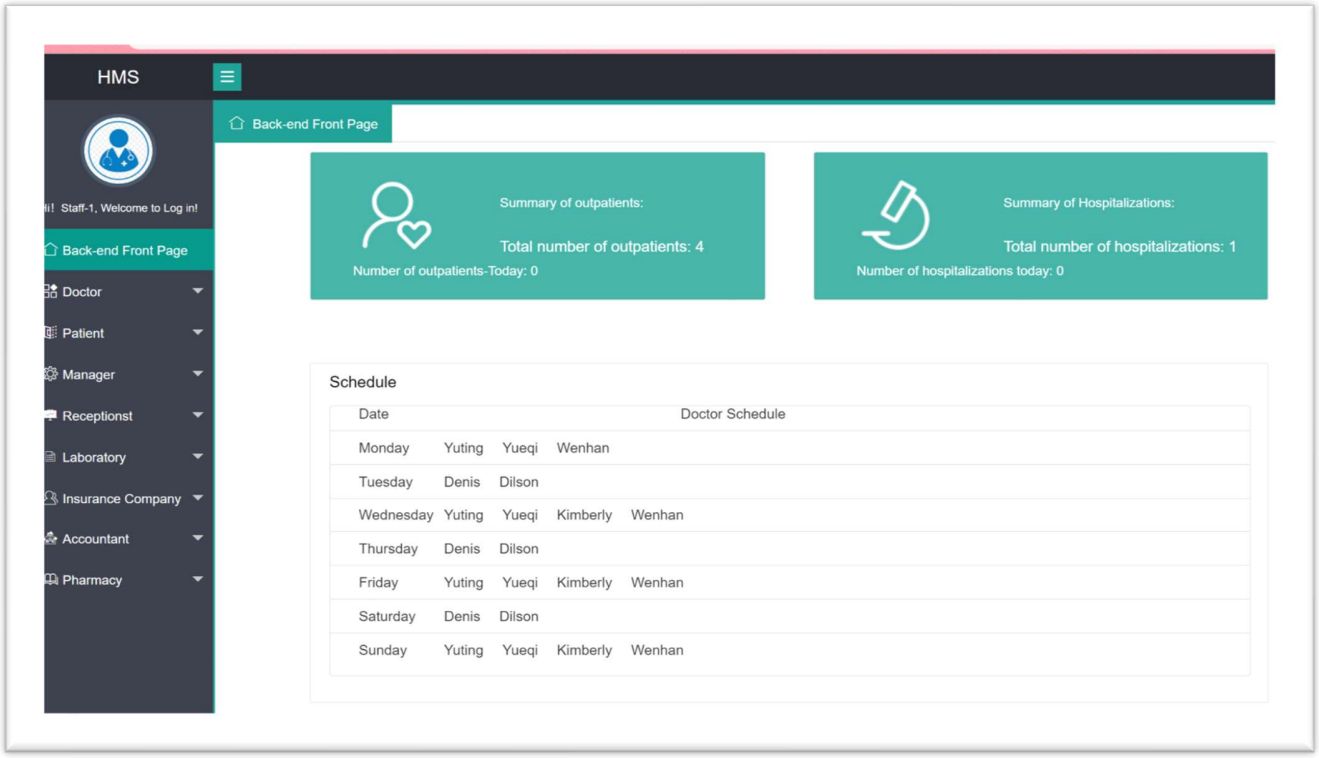
This implementation is a preliminary demonstration of the simple blueprint plan for this system.Which includes 3 main frameworks on this WebApp: **SpringBoot** +**layui+shiro**.Use **Java and Javascript** as programming languages. The database is constructed by **MySQL**, ran on the Navicat platform to modify some features. This demo shows some website functions according to the Use-Case diagram and Database Design diagram to deploy.

Here are some simple demo screenshots:

1)Login page：

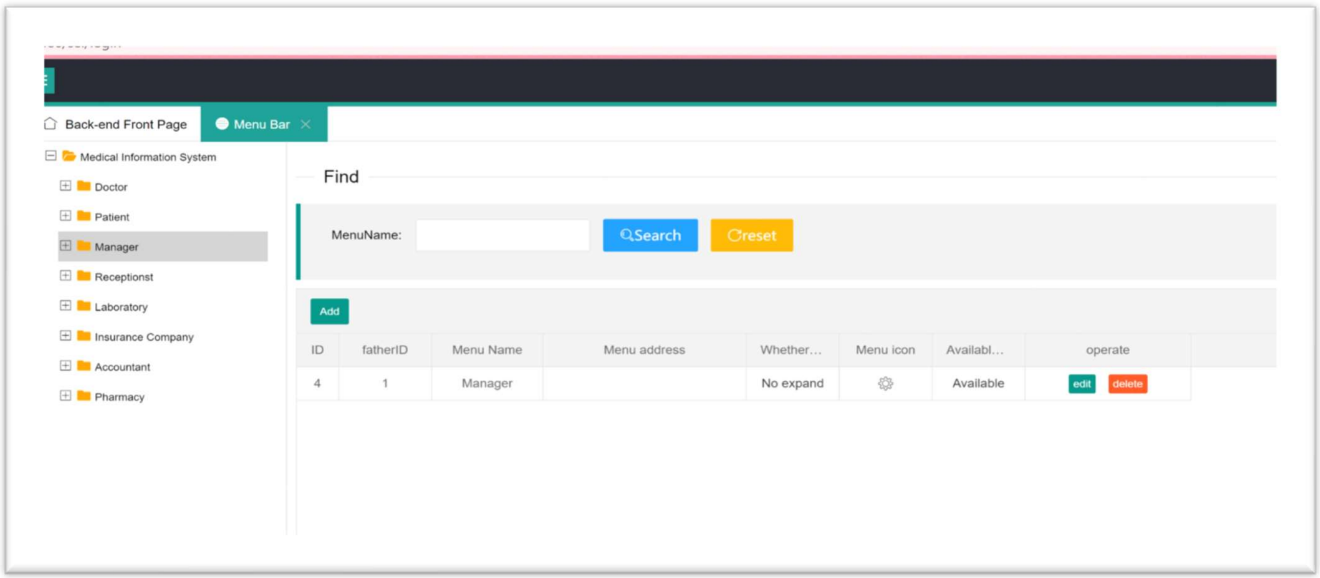


2)Navigation Page:(following the Use-Case diagram)



3)Instance Feature Demonstration:

This screenshot shows how the manager revises important medical information on the backend.



5. Conclusion

Working on this project, we found out how to utilize what we've learned in INSY5341 course and practiced it by building the information system of our choice.

During the Planning phase we created a System proposal that included System Request, Feasibility Analysis and Requirements Definition. This part helped us better understand the business side of the system, why the system was needed.

In the Analysis phase we further elaborated our systems in Activity Diagram, Use Case Diagram, Use Case Descriptions, Initial Class Diagram and Sequence Diagrams. Here we could better understand the participants of our system and how they are connected with each other. Hands-on experience with UML helped us define correctly actors, objects, associations and other elements of diagrams.

In Design phase we moved closer to programming. We revised Class Diagram and created Final Class Diagram on its basis, grouped classes into packages in Package diagram, created Database design and Data Access and Manipulation Design. In this part we looked at the system through the eyes of a software developer and created a map of how to create the actual code.

The Implementation phase was our humble attempt to create the interface of Medical Information System, how we thought it should look like.

All in all, this project helped us consolidate our knowledge of system analysis and design, reveal misunderstanding of some concepts and fill in these gaps. We also learned how to work better in a team environment.