# Design Document

Software Design Practices - CS 753

Under the guidance of

Prof. Sujit Kumar Chakrabrati

# Pharmacy Management System

MT2020004 : Manu Dandotiya

MT2020017 : Meghna Dubey

MT2020056 : Chandrika Bhuyan

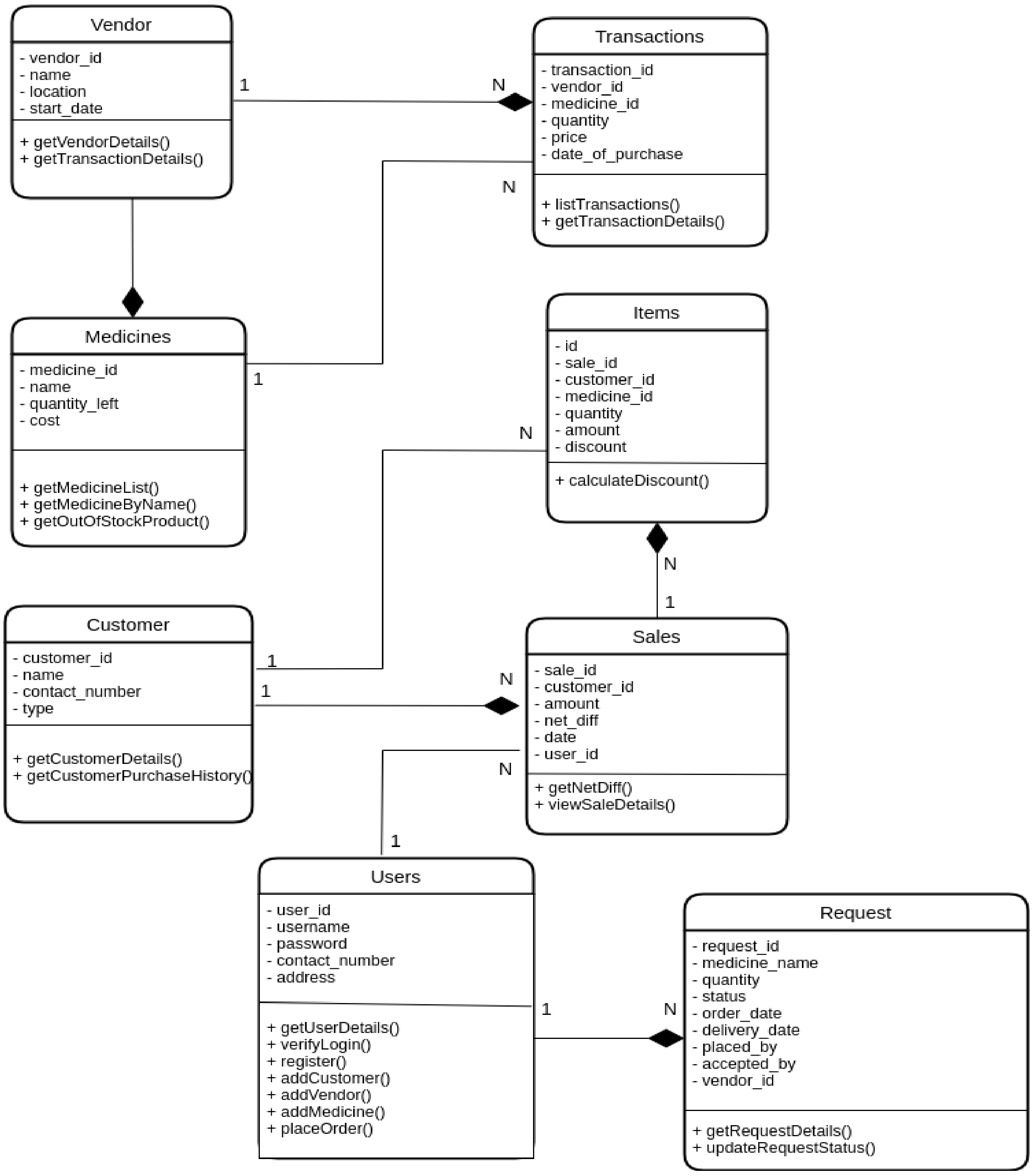
MT2020064 : Ganesh Bhat

MT2020103 : Nupur Banerjee

## Contents

# 1. Structural Modelling

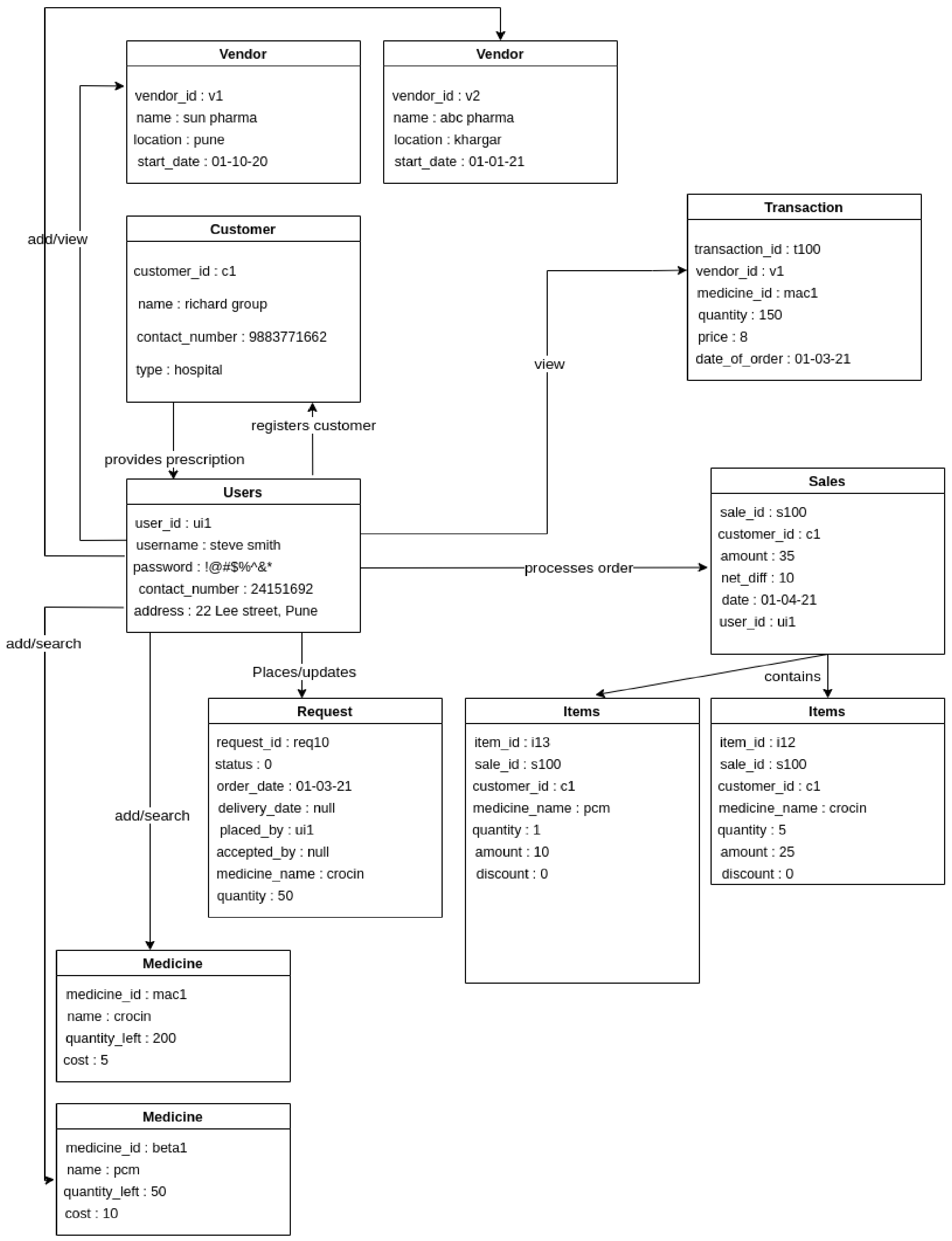
## 1.1 Class Diagram



The various classes involved in the application along with their mapping are:

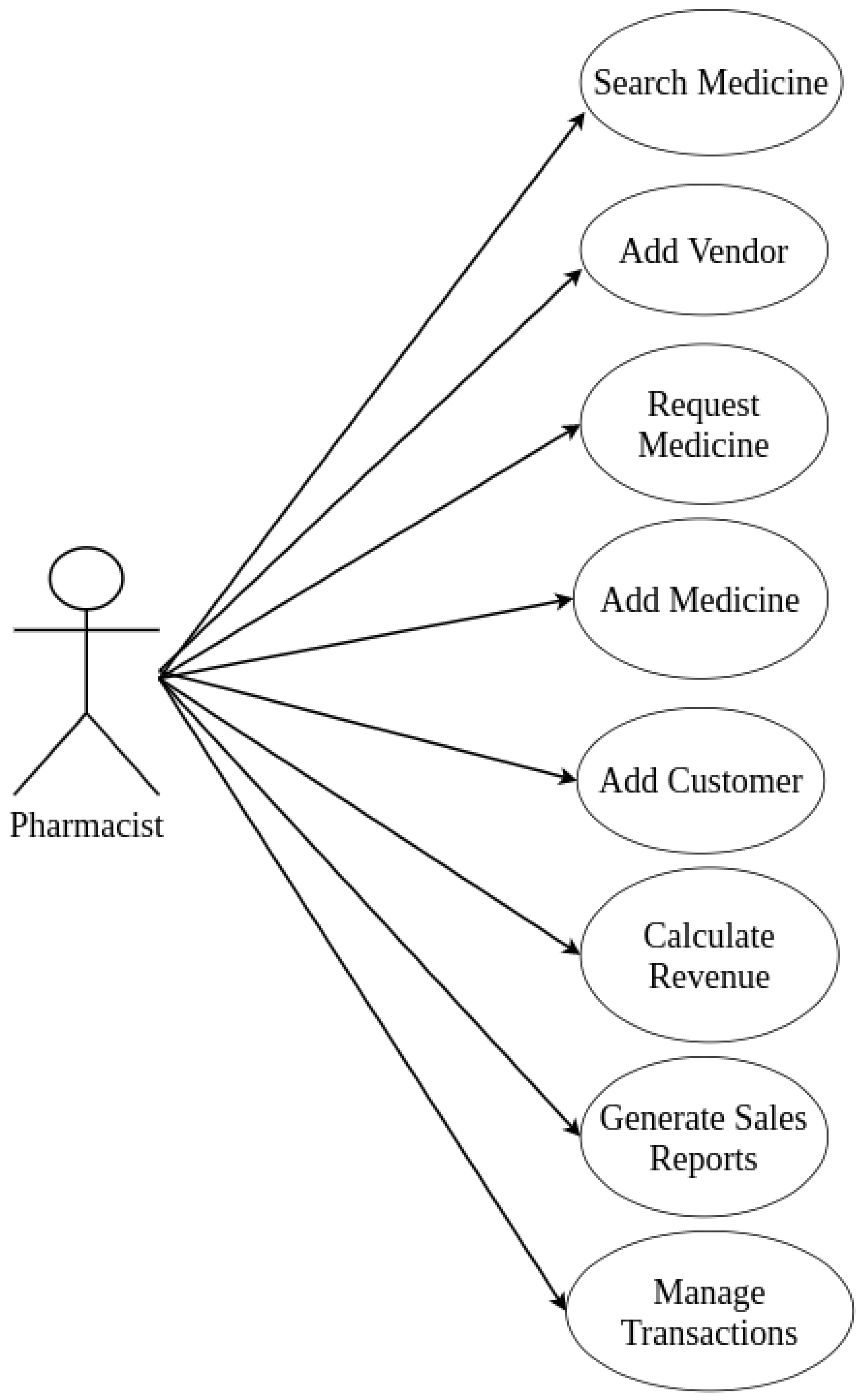
- **Vendor, Transaction** : A vendor can be involved in multiple transactions with the pharmacy.
- **Vendor, Medicine** : Medicine cannot exist without the vendor, since they will be supplying it to the pharmacy.
- **Medicine, Transaction** : The same medicine can be bought multiple times.
- **Customer, Items** : A customer can ask for a list of items from the pharmacist.
- **Customer, Sales** : Sales is the consolidated summary of a customer's order.
- **Sales, Items** : A sale can contain multiple items based upon the customer's request.
- **Users, Sales** : A pharamcist can process multiple sales.
- **Users, Request** : Pharmacists can place request for medicines as and when required.

## 1.2 Object Diagram



# 2. Dynamic Modelling

## 2.1 Use Case Diagram

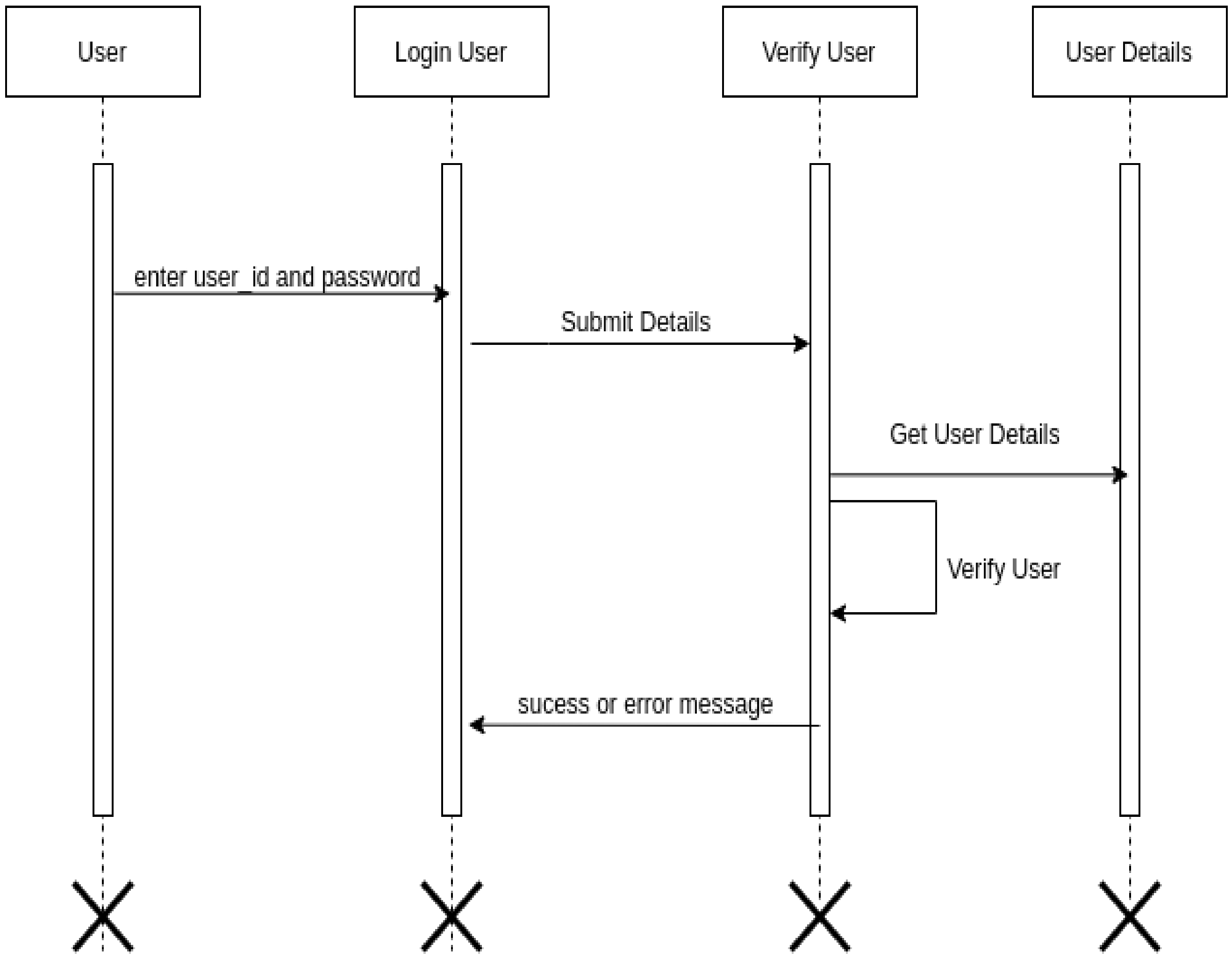


The main user of the system, i.e, the pharmacist can perform the following functions:

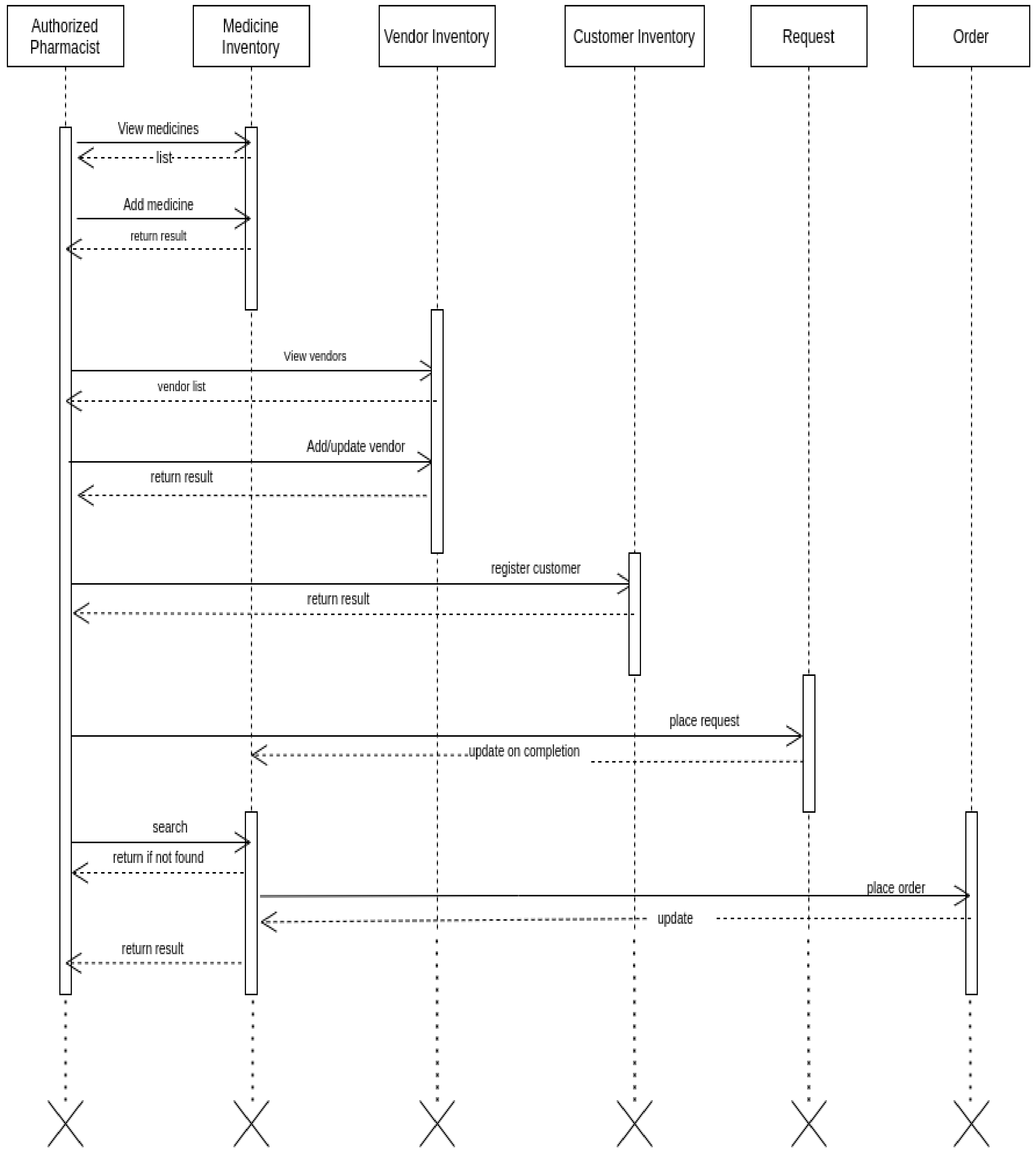
- Search for medicines from the inventory.
- Add vendors to the vendor inventory.
- Add/Update medicines in the medicine inventory.
- Place request for medicines that are either out of stock or are less in quantity.
- Register new customers.
- Generate bill for the customer.
- View transactions done with the vendors.
- Calculate gross different for the pharmacy (profit/loss).

## 2.2 Sequence/Collaboration diagrams

- Sequence diagram for user login

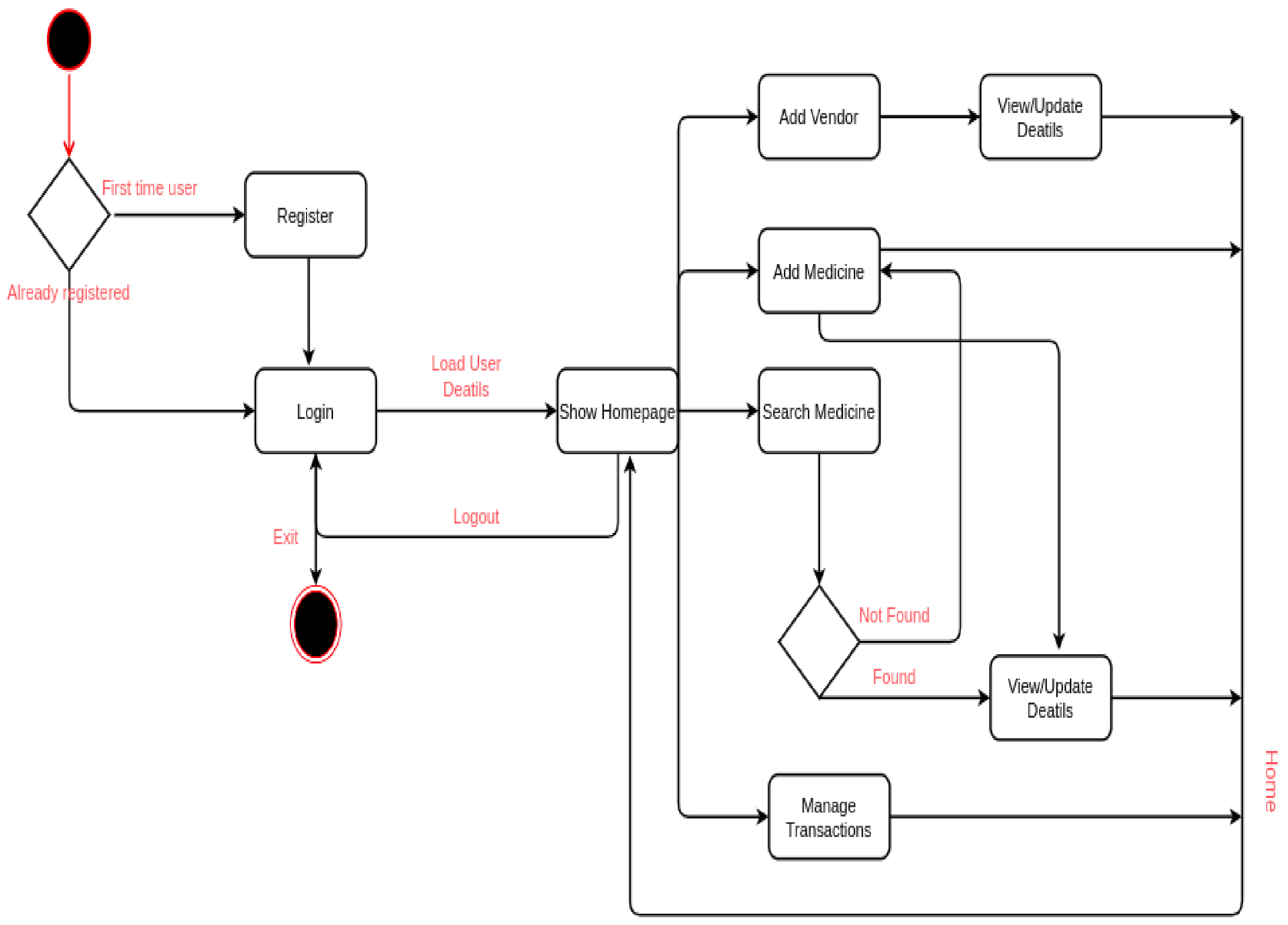


- Sequence diagram for general application functionalities

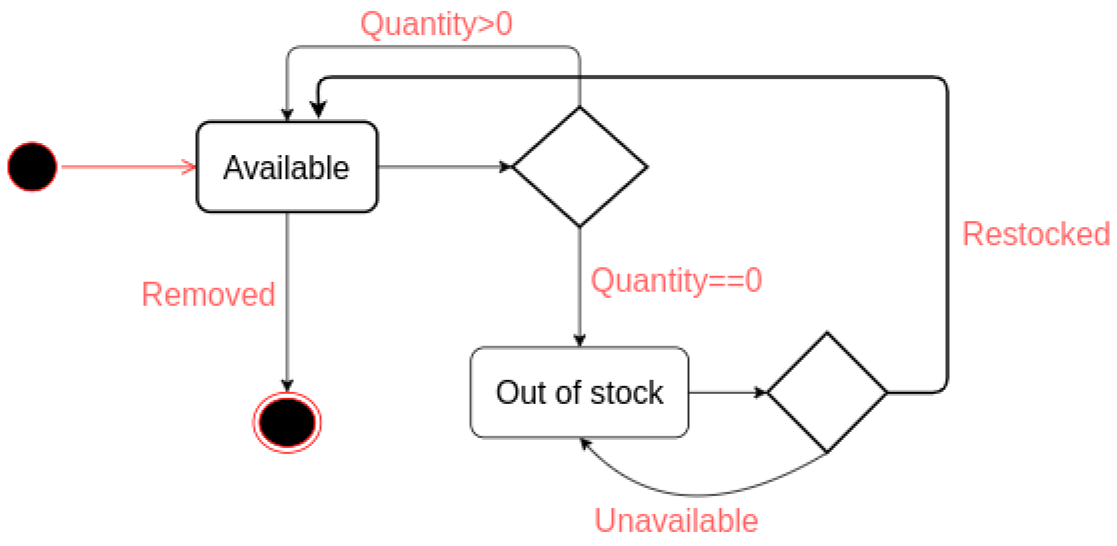


## 2.3 State Diagram

- State Diagram for pharmacist



- State diagram for medicine stock

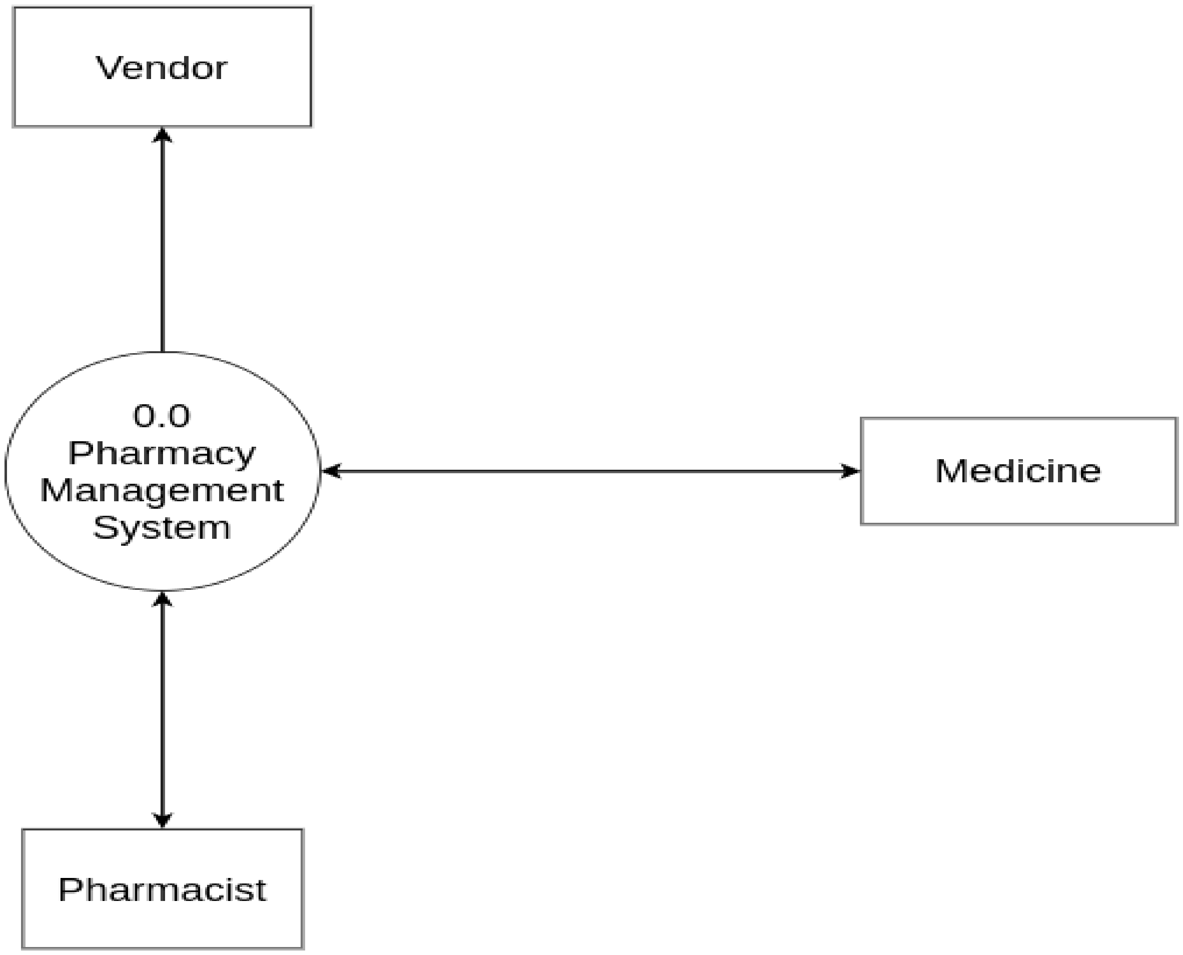


# 3. Functional modelling

## 3.1 Data Flow Diagram

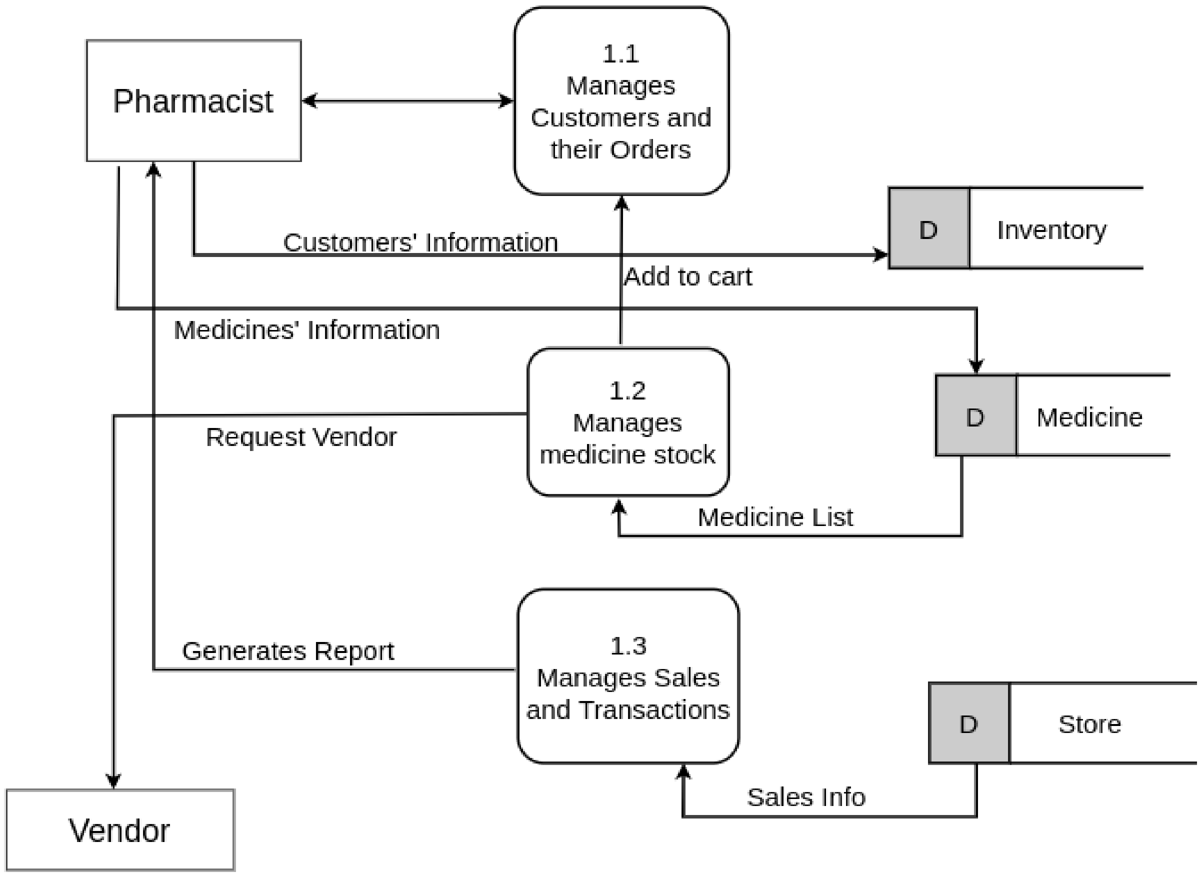
**Level 0 DFD :**

- It is also known as context diagram.
- It's supposed to be an abstract view, with the mechanism represented as a single process with external parties.



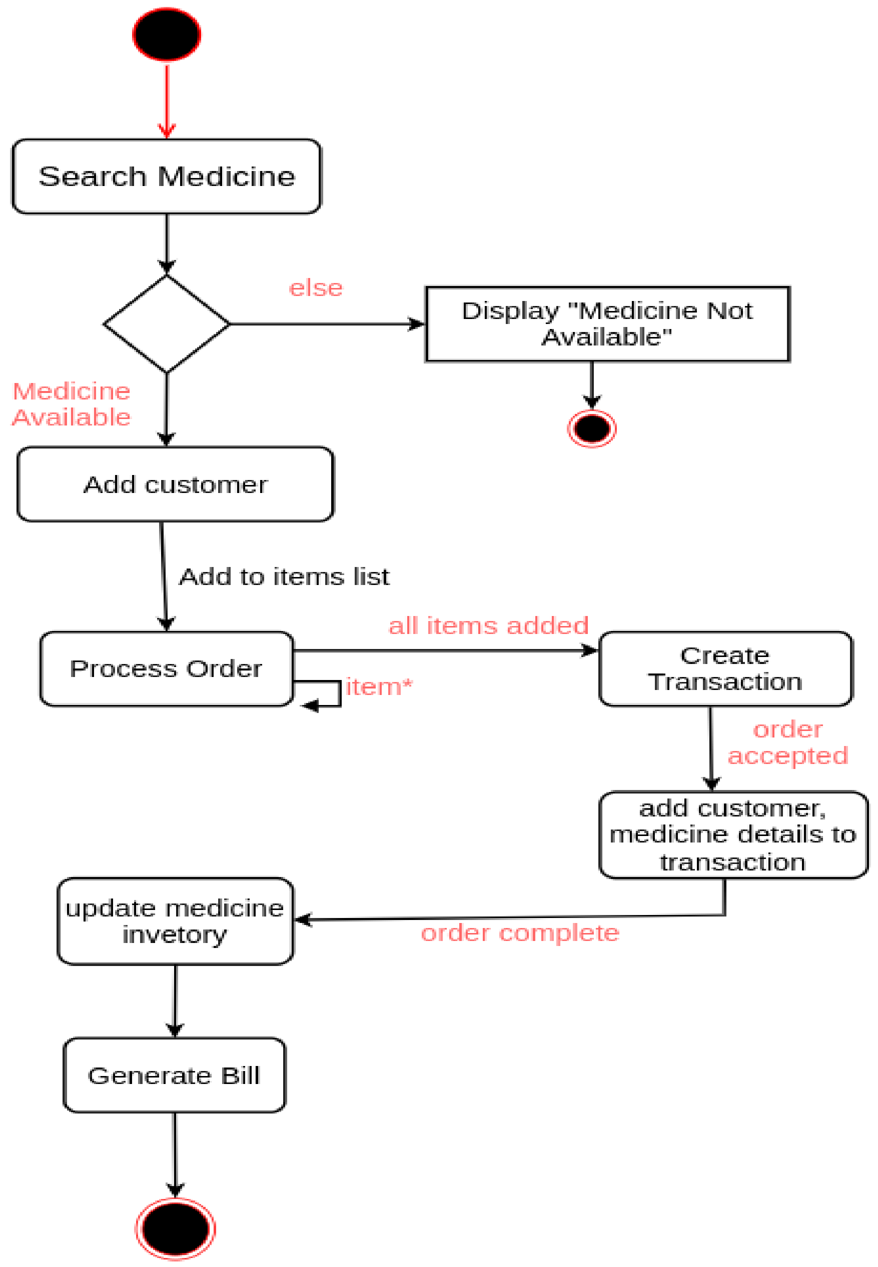
**Level 1 DFD :**

- In this level, the system must display or reveal further processing information.
- The following are essential data to accommodate:
    - Customer Information
    - Vendor Information
    - Sales
    - Stock record
    - Medicine Information

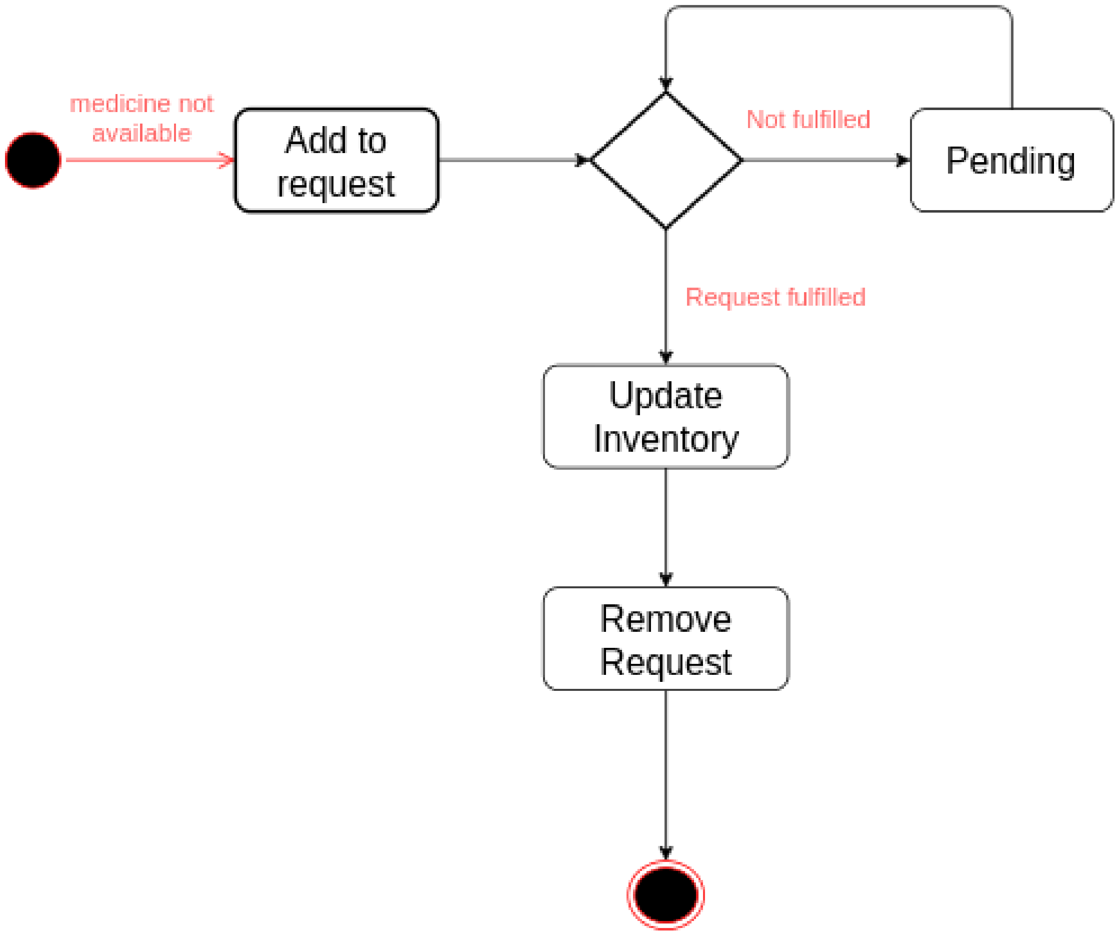


## 3.2 Activity Diagram

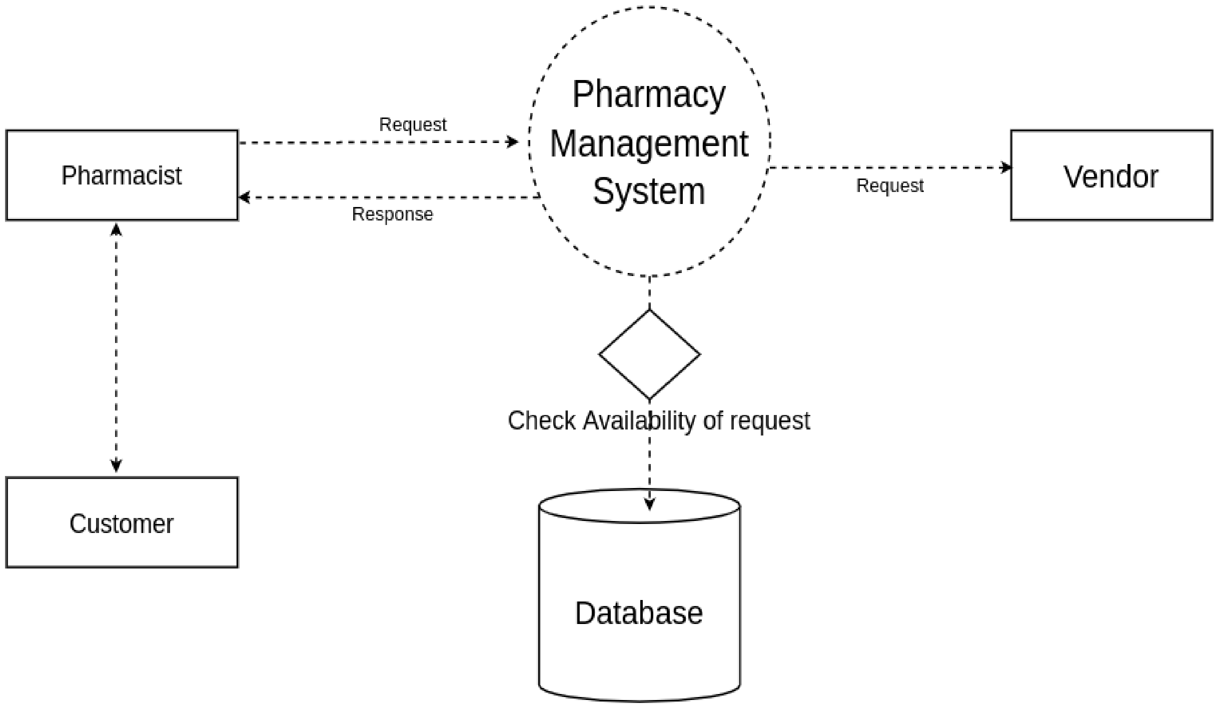
- Activity diagram for placing an order



- Activity diagram for placing request for medicines.

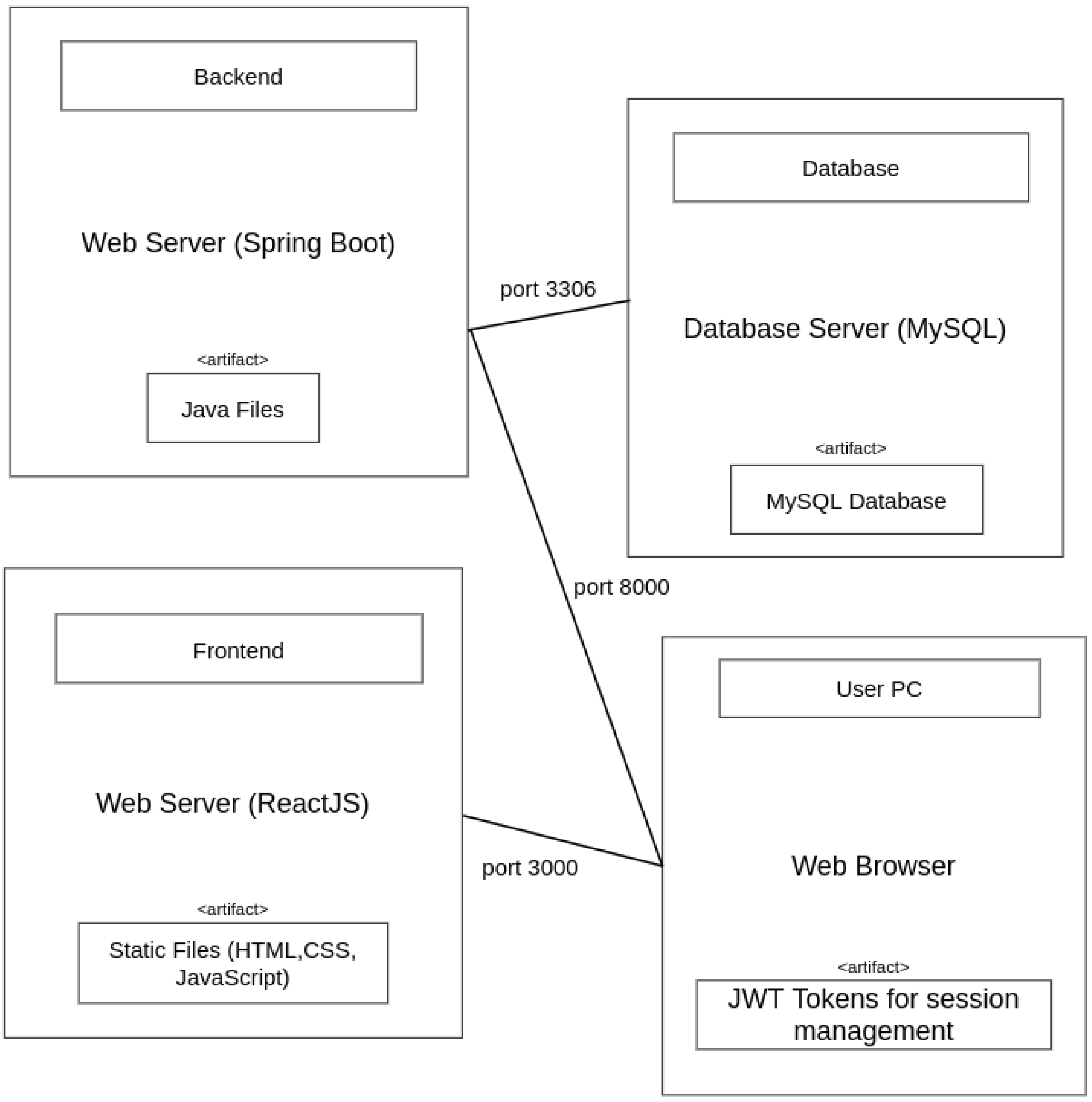


## 4. Architectural diagrams



The customer will provide a prescription or list of medicines required to the pharmacist. The pharmacist will then query the system and process the order as per the availability.

The pharmacist will also place requests for medicines that are required and update their status as and when the requests are fulfilled.

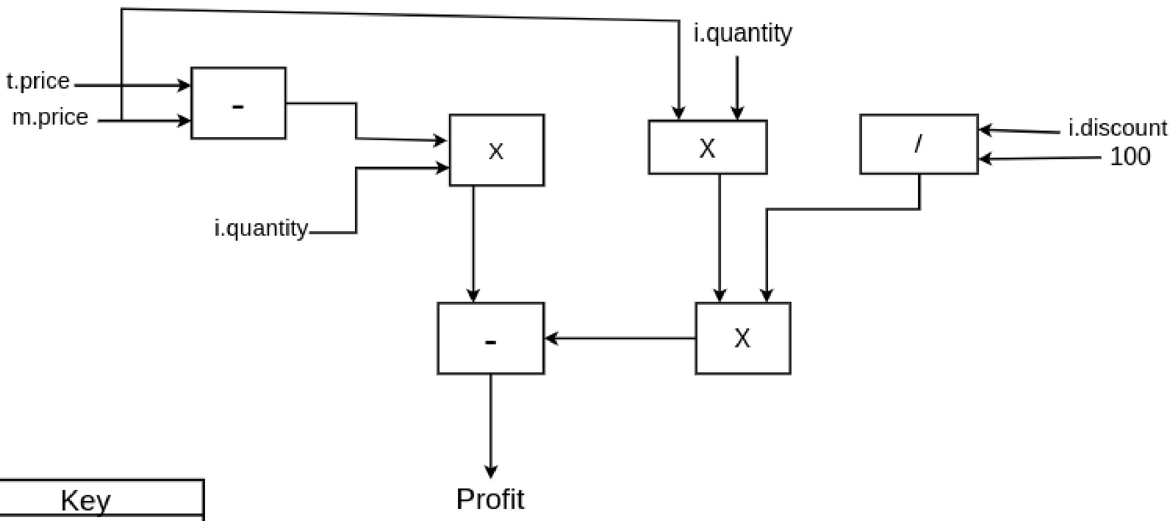


The tools that will be used in the application are:

- Backend : SpringBoot
- Database : MySQL
- Frontend : ReactJS
- JWT tokens for managing user session

# 5. Mathematical Modelling

## 5.1 Block Diagram



| Key |
|---|
| t : transactions |
| m : medicine |
| i : item |

| Formula |
|---|
| total_discount = selling price * quantity * discount / 100<br>Profit = (Cost Price - Selling Price) * quatity - total_discount |

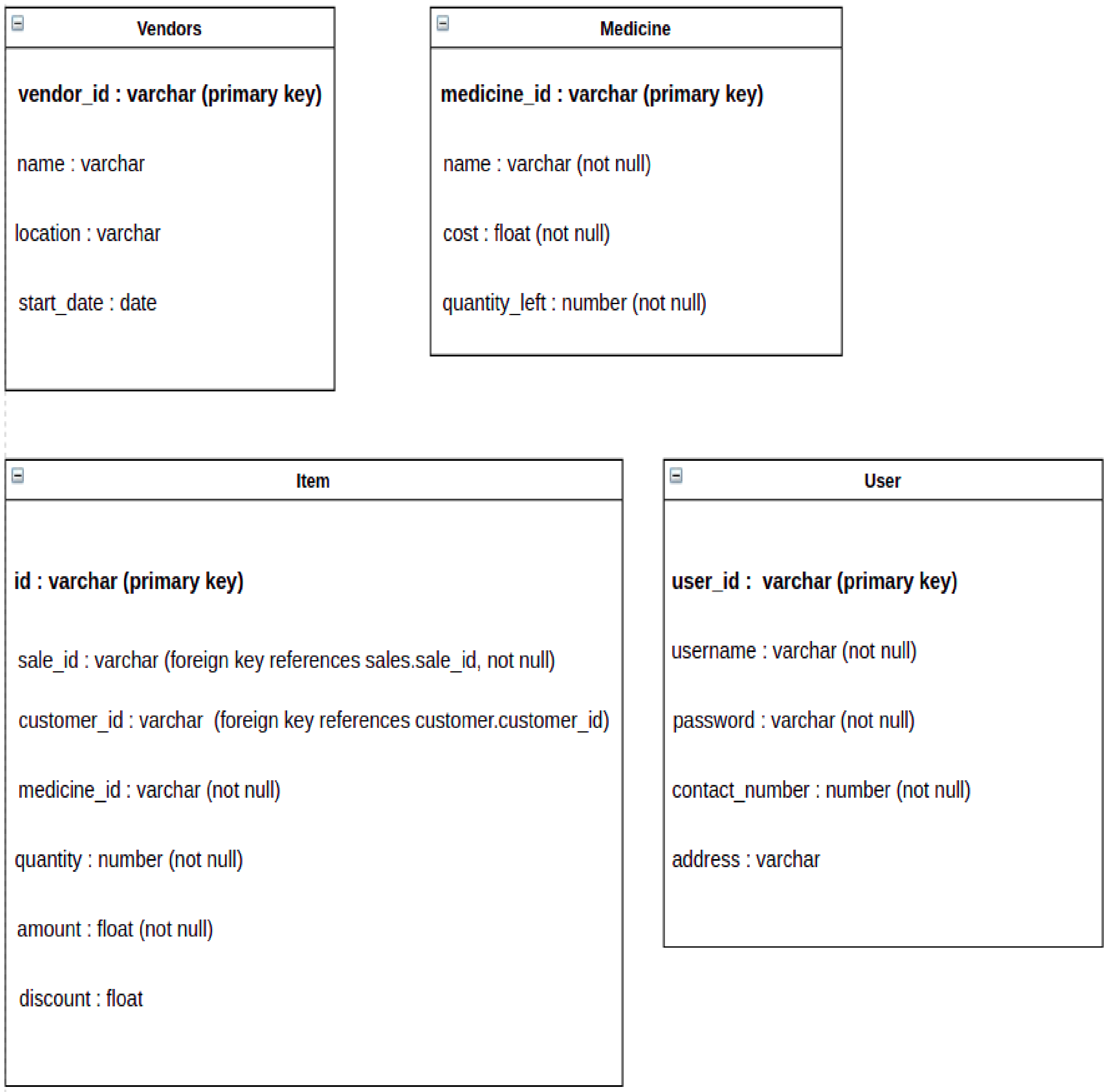
- transactions.price is the price at which the pharmacist bought the respective medicine (cost price).

- medicine.price is the price at which the medicine will be sold to the customer (selling price).

the net profit on the sale of a particular item is calculated as :

*total_discount = selling price * quantity * discount / 100*
*total_amount = (cost price - selling price)* quantity*
*profit = total_amount - total_discount*

# 6. Data Model

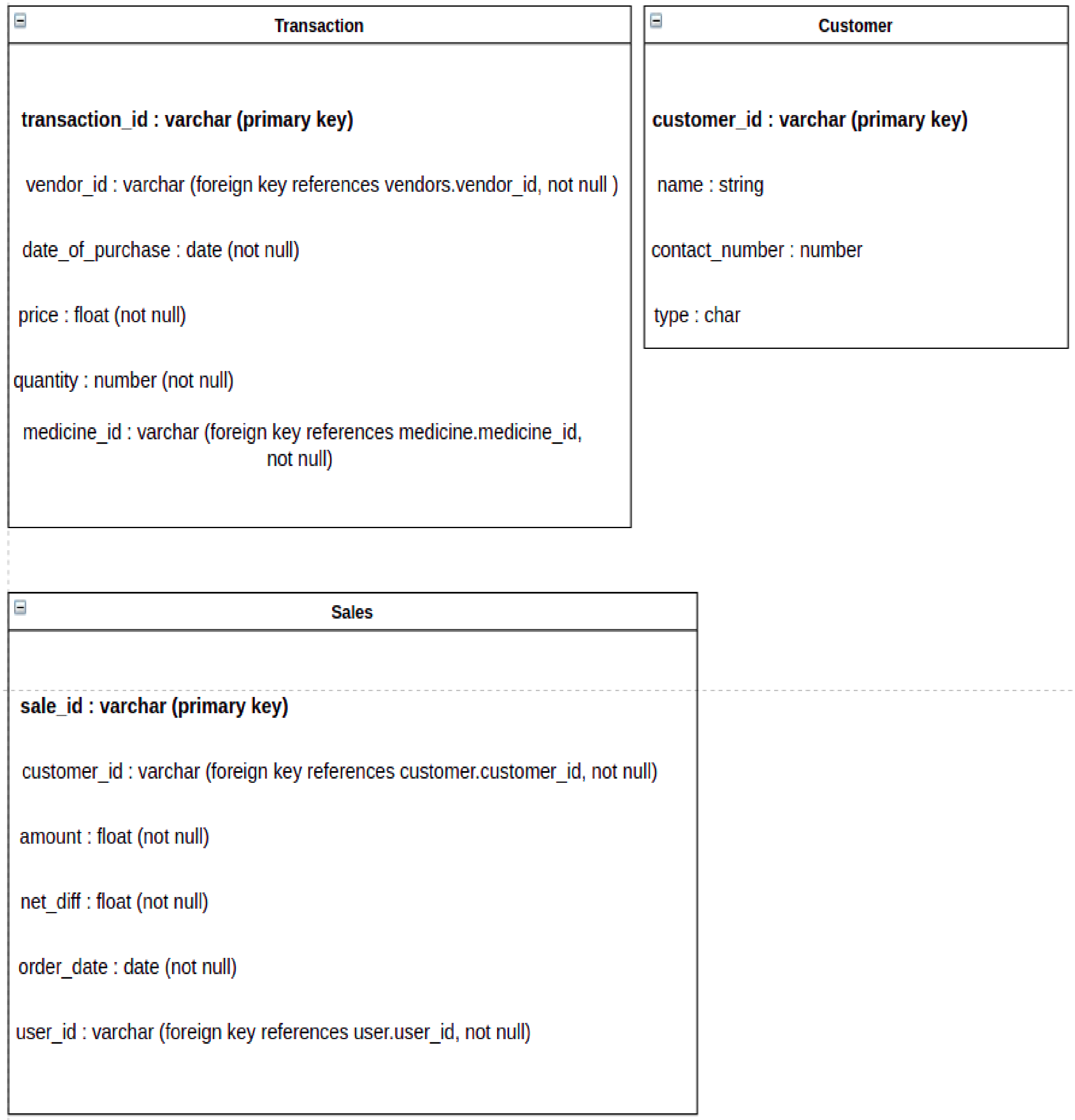


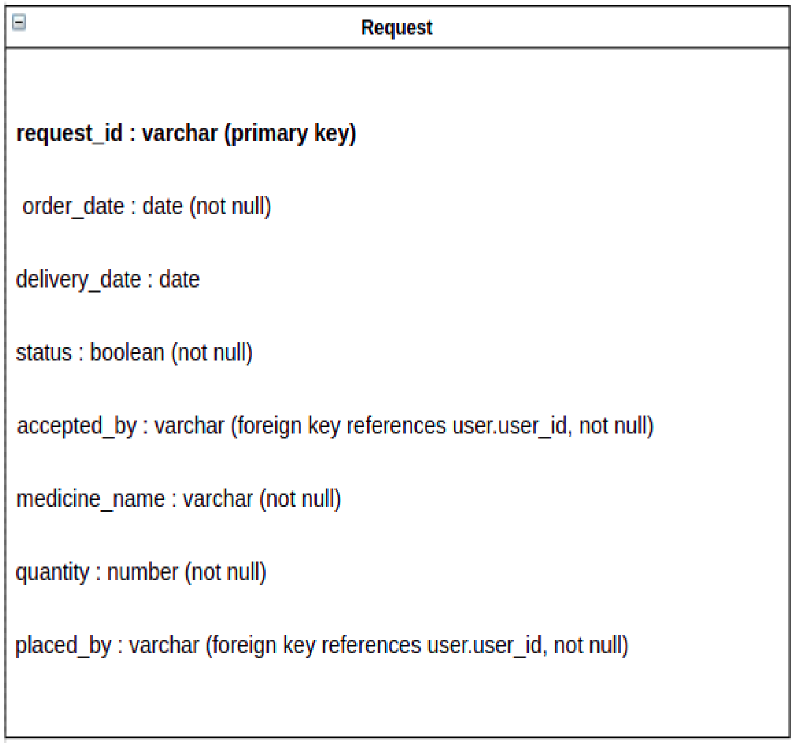
| Vendors |
|---|
| **vendor_id : varchar (primary key)** |
| name : varchar |
| location : varchar |
| start_date : date |

| Medicine |
|---|
| **medicine_id : varchar (primary key)** |
| name : varchar (not null) |
| cost : float (not null) |
| quantity_left : number (not null) |

| Item |
|---|
| **id : varchar (primary key)** |
| sale_id : varchar (foreign key references sales.sale_id, not null) |
| customer_id : varchar (foreign key references customer.customer_id) |
| medicine_id : varchar (not null) |
| quantity : number (not null) |
| amount : float (not null) |
| discount : float |

| User |
|---|
| **user_id : varchar (primary key)** |
| username : varchar (not null) |
| password : varchar (not null) |
| contact_number : number (not null) |
| address : varchar |


Transaction
transaction_id : varchar (primary key)
vendor_id : varchar (foreign key references vendors.vendor_id, not null )
date_of_purchase : date (not null)
price : float (not null)
quantity : number (not null)
medicine_id : varchar (foreign key references medicine.medicine_id, not null)
Customer
customer_id : varchar (primary key)
name : string
contact_number : number
type : char
Sales
sale_id : varchar (primary key)
customer_id : varchar (foreign key references customer.customer_id, not null)
amount : float (not null)
net_diff : float (not null)
order_date : date (not null)
user_id : varchar (foreign key references user.user_id, not null)

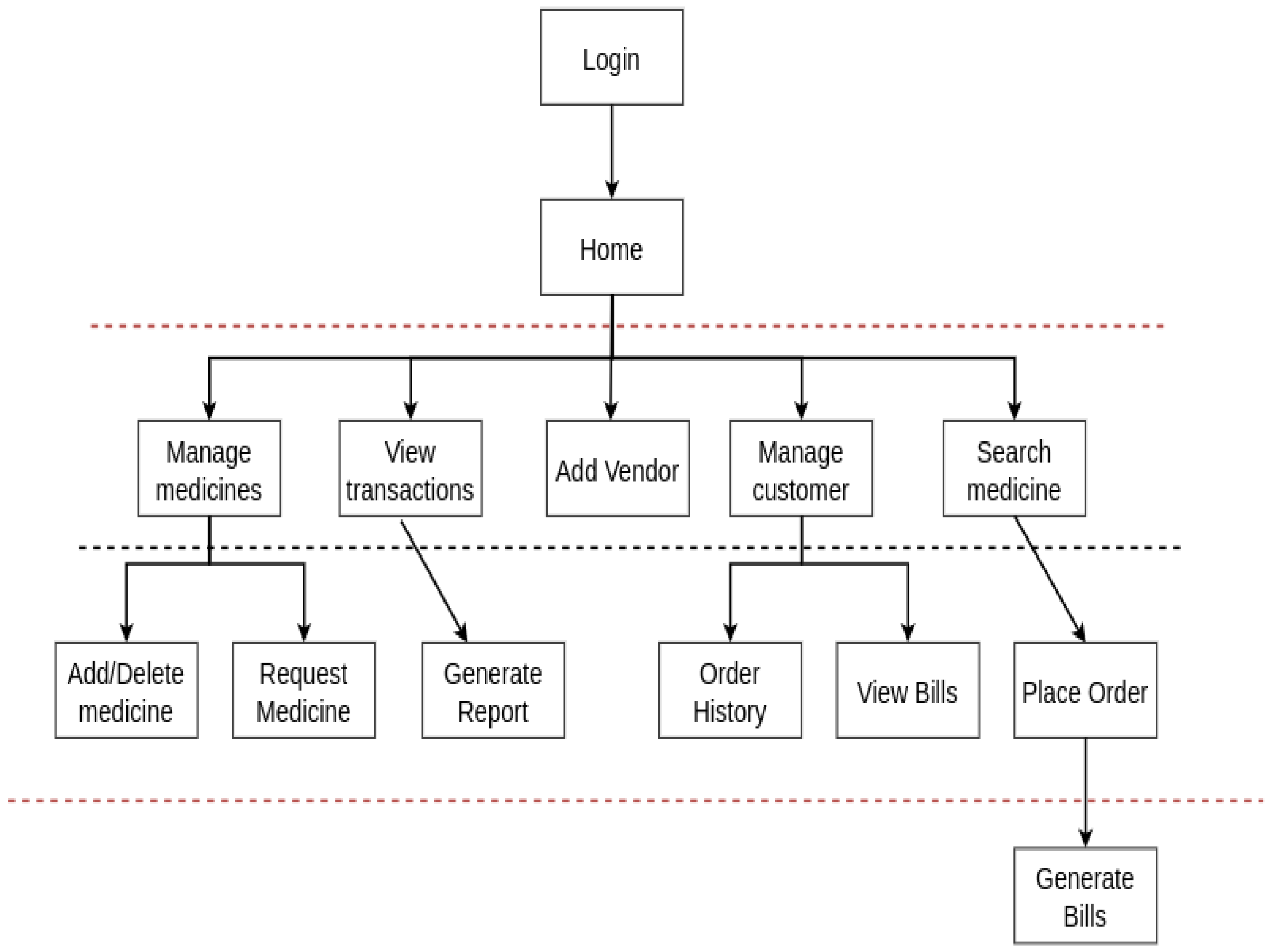

Request
request_id : varchar (primary key)
order_date : date (not null)
delivery_date : date
status : boolean (not null)
accepted_by : varchar (foreign key references user.user_id, not null)
medicine_name : varchar (not null)
quantity : number (not null)
placed_by : varchar (foreign key references user.user_id, not null)

The tables required for the application are :

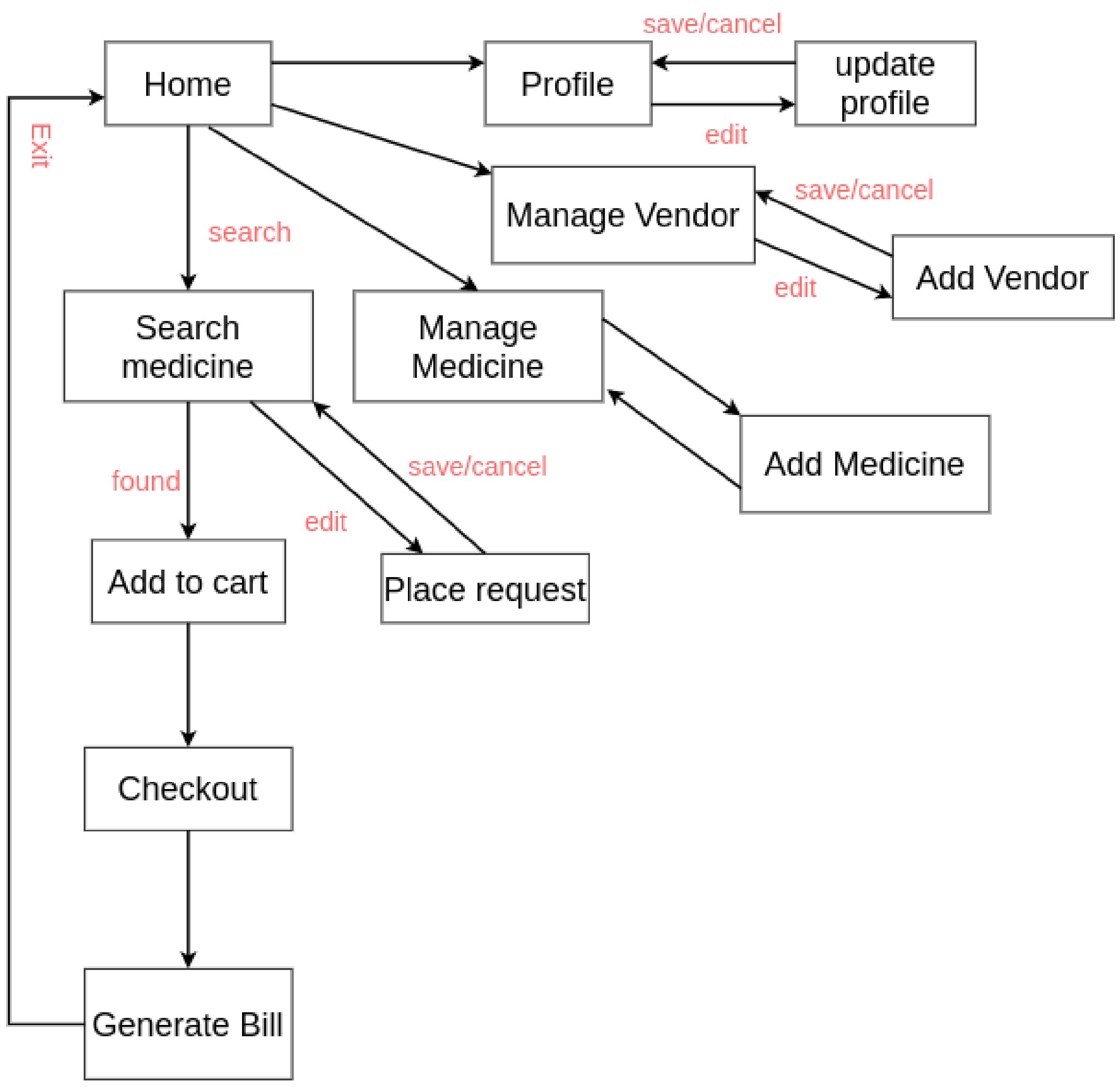
- **Vendors**: this will store the details of the vendor that will supply medicines to the pharmacy.
- **Medicine**: this will contain a list of medicines and its details.
- **User**: this contains the login information and personal details of the pharmacists.
- **Customer**: this contains details about the customer which can be of 2 types - hospital, individual.
- **Request**: this will keep track of the medicines for which request has been placed (that will be fulfilled by the vendor).
- **Transaction**: this will store details about the transaction made between the pharmacy and the vendor.
- **Sales**: this will store details related to a particular order entered by the pharmacist.
- **Item**: this will store details of individual item related to a particular sale order.

# 7. UI Design

## 7.1 Hierarchy of Screens

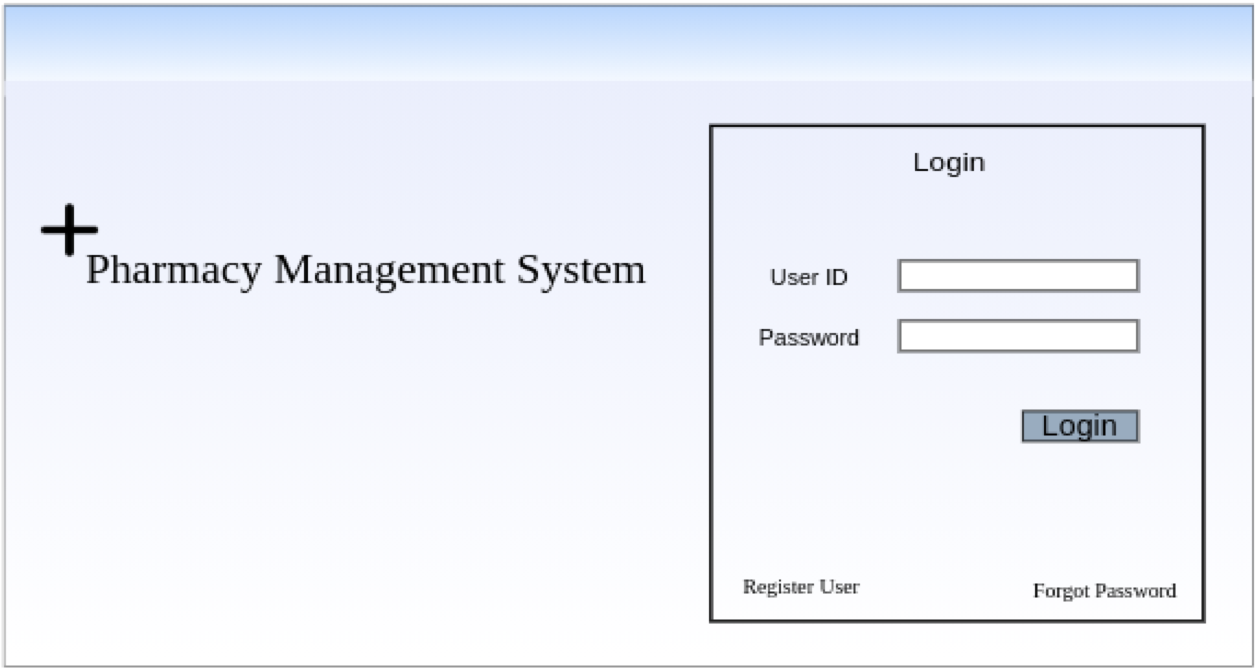


## 7.2 Navigational Diagram

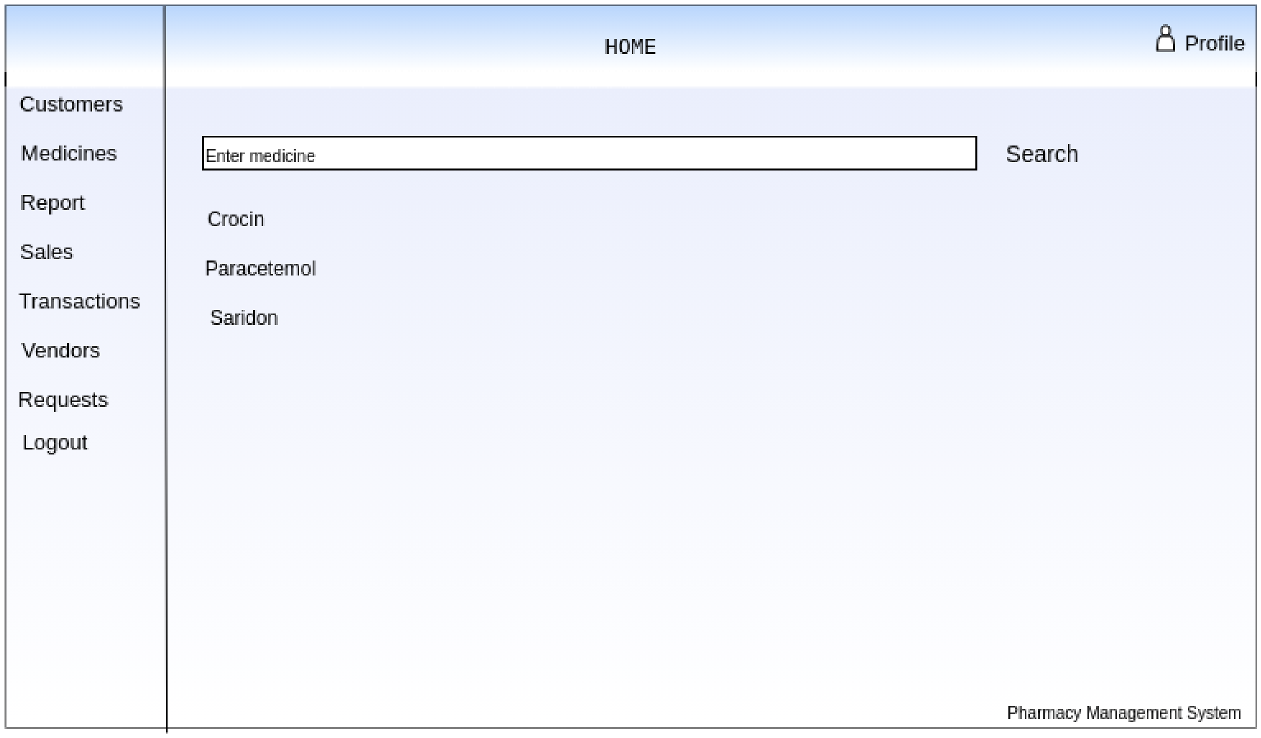


## 7.3 Screen Sketches

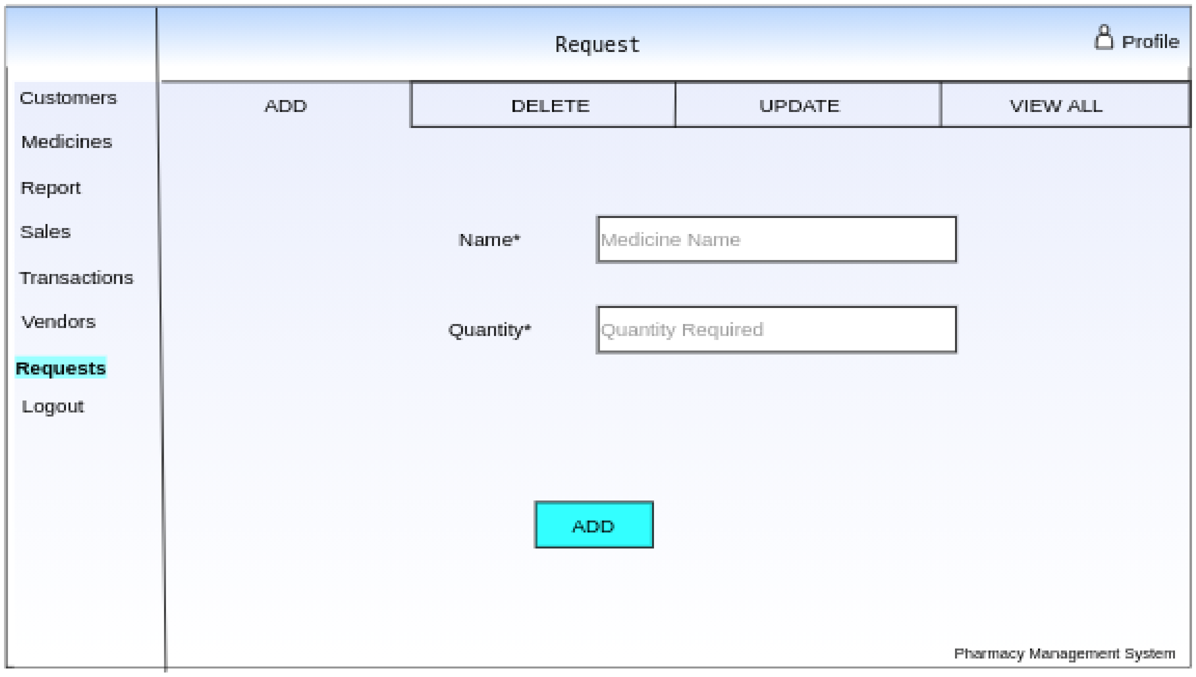
### 7.3.1 Login Page



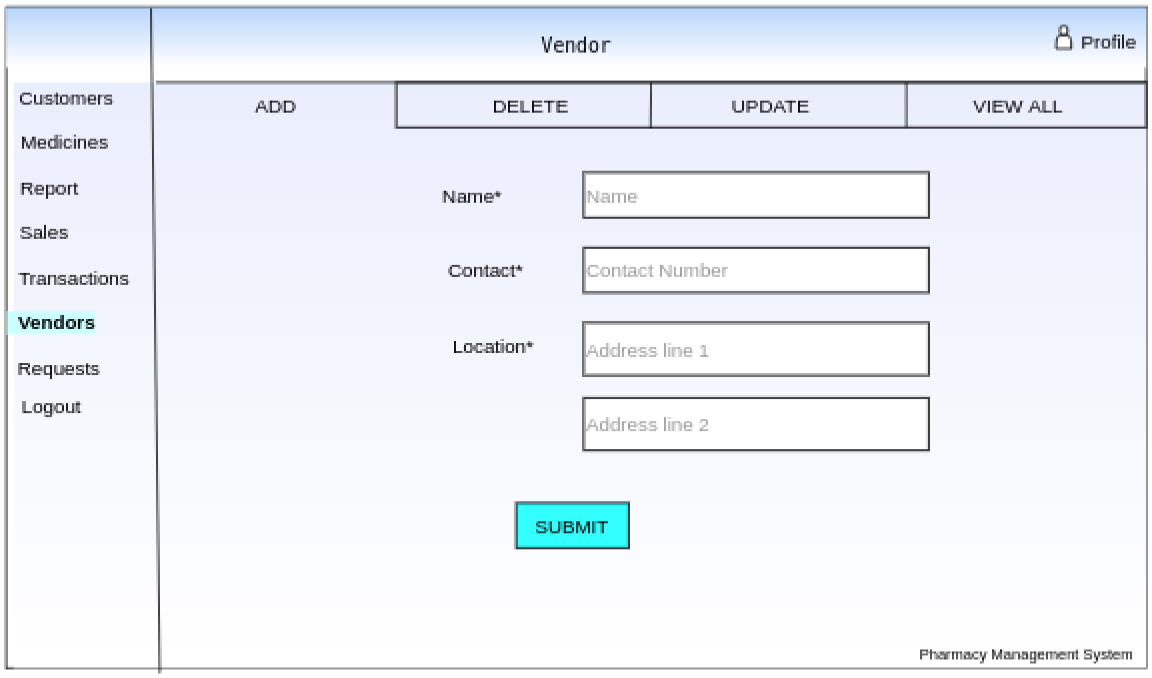
### 7.3.2 Home Page



### 7.3.3 Place request for medicines



### 7.3.4 Add vendor details



## 8. Non-Functional Requirements

1. Usability: We are using react to build a single page app which will contribute to an intuitive user journey.
2. Performance:
    a. By making it a single page app using react we get a performance boost as the majority of application resources are loaded once.
    b. To create a customer's bill we are not creating another table instead of that we are creating views which will be a space saver.
    c. We will be handing the majority of computational operations e.g avg , sum , percentage in the high level language instead of doing them by SQL query.
3. Testability: We are making the code loosely coupled with each other so that each component can be easily testable individually.
4. Maintainability: We are following a clean and consistent coding standard , with human readable and sensible names of methods , variables and classes and tried to achieve minimum redundancies in the code base.