###### 

PUCIT

Punjab University College of Information Technology

**Second Deliverable for Object Oriented Approach**

**Version 1.0**

Table of Contents

[1 Introduction 3](#_Toc148483291)

[1.1 Usecase descriptions 3](#_Toc148483292)

[1.2 Use Case Diagram (Refined) 4](#_Toc148483293)

[1.3 Domain Model 4](#_Toc148483293)

[1.4 Sequence Diagram 6](#_Toc148483295)

[1.4.1. Defining a Sequence diagram 6](#_Toc148483296)

[1.4.2. Basic Sequence Diagram Symbols and Notations 6](#_Toc148483297)

[1.4.3 Example 10](#_Toc148483298)

[1.4.4 Distributing Control Flow in Sequence Diagrams 10](#_Toc148483299)

[1.5 Collaboration Diagram 13](#_Toc148483300)

[1.5.1 Contents of Collaboration Diagrams 14](#_Toc148483301)

[1.5.2 Constructs of Collaboration Diagram: 14](#_Toc148483302)

[1.6 Operation Contracts 15](#_Toc148483303)

[1.7 Design Class Diagram 16](#_Toc148483304)

[1.7.1 Create Initial Design Classes 16](#_Toc148483305)

[1.7.2 Designing Boundary Classes 17](#_Toc148483306)

[1.7.3 Designing Entity Classes 17](#_Toc148483307)

[1.7.4 Designing Control Classes 17](#_Toc148483308)

[1.7.5 Identify Persistent Classes 18](#_Toc148483309)

[1.7.6 Define Class Visibility 19](#_Toc148483310)

[1.7.11 Design Class Relationships 23](#_Toc148483311)

[1.8 Data Model 27](#_Toc148483313)

# 

# 

## 1 Introduction

Third deliverable is all about the usecase modeling and software design. In the previous deliverable, analysis of the system is completed. So we understand the current situation of the problem domain. Now we are ready to strive for a solution for the problem domain by using object-oriented approach. Following artifacts must be included in this deliverable.

1. Use case description
2. Use case diagram refined
3. Domain Model
4. Sequence Diagram
5. Collaboration Diagram
6. Operation Contracts
7. Design Class Diagram
8. Data Model

Now we discuss these artifacts one by one as follows:

## 1.1 Usecase Description

**UC #1:**

**Log in:**

|  |  |  |
| --- | --- | --- |
| Use-case Number | UC #1 | |
| Use-Case Name | Log in | |
| Priority | High | |
| Actor | Customer | |
| Description | This use case describes how Customer to login into Womanista. | |
| Precondition | Customer must be woman | |
| Post-condition | If the use case was successful, the actor is now logged into the Womanista. If not, the app state is unchanged. | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The customer is on the home page to login to the app. 2. The customer enters email and password, Click on Login Button. | 1. The App promotes the customer to enter Email, Password. 2. The admin verifies that all the filled have been filled out and valid. 3. The app successfully logged in the app. 4. Use case Exit |
| Alternate Scenario | 6.1 If all fields are not filled out and not matched to the email and password the app notifies the actor a message Verify Email or Password and then goes back or returns to step 4 of Main Success Scenario to enter again. | |

**UC #2:**

**Log out:**

|  |  |  |
| --- | --- | --- |
| Use-Case Number | UC #2 | |
| Use-Case Name | Log out | |
| Priority | High | |
| Actor | Customer | |
| Description | These use case allow Customer to log out from the App. | |
| Precondition | UC #1 | |
| Post Condition | App logs out | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The Customer wants to log out 2. The Customer clicks the log out button | 1. The app responds to the requested action. 2. The app displays a message that the Customer logged out from the app. 3. Use case Ends |

**UC #3:**

**Authentication:**

|  |  |  |
| --- | --- | --- |
| Use-case Number | UC #4 | |
| Use-Case Name | Authentication | |
| Priority | High | |
| Actor | Admin | |
| Description | This use case describes allows the admin to Authenticate the customer using email. | |
| Precondition | Customer cannot use the app if Authentication failed. | |
| Post-condition | If the use case was successful, the customer can now use the Womanista. If not, the app state is unchanged. | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The admin is on the admin dashboard to Authenticate the customer using email and CNIC customer to the app. 2. The admin clicks on Authenticate Button | 1. The customer can now use the Womanista 2. The app shows Authenticated successfully message to the customer 3. Use case Exit. |
| Alternate Scenario | If CNIC is not valid or does not exist the app shows are not filled out and not matched CNIC the app notifies the admin a message Authenticate CNIC and then goes back or returns to step 1 of Main Success Scenario to enter again. | |

**UC #4:**

**Cab Booking:**

|  |  |  |
| --- | --- | --- |
| Use-case Number | UC #4 | |
| Use-Case Name | Cab Booking | |
| Priority | Medium | |
| Actor | Customer, Drivers | |
| Description | This use case describes allows the customer to cab booking with female driver | |
| Precondition |
| Post-condition | If the use case was successful, the customer can now use the Womanista. If not, the app state is unchanged. | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The customer can book a car Enters an address and email using email and location of customer to the app. 2. The admin clicks on Book Button | 1. The customer book cab can now use the Womanista 2. The app shows cab booked message to the customer 3. Use case Exit. |
| Alternate Scenario | If email and location is not valid or does not exist the app shows are not filled out and not matched email and location the app notifies the customer CAB NO BOOK message and then goes back or returns to step 1 of Main Success Scenario to enter again. | |

**UC #5:**

**Search Products:**

|  |  |  |
| --- | --- | --- |
| Use-case Number | UC #5 | |
| Use-Case Name | Search Products | |
| Priority | Low | |
| Actor | Customer | |
| Description | This use case describes allows the customer to search self-defense training tools and Sanitary products by choosing from the list | |
| Precondition | UC #1,3,4 | |
| Post-condition | If the use case was successful, the customer can search self-defense tools by using Womanista. If not, the app state is unchanged. | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The customer enters the name of the Self Defense and Sanitary products 2. Selects from the drop down 3. Enter address for delivery 4. Select Search Button | 1. Shows and drop-down menu of the products 2. App shows the product and its price 3. Use case Exit. |
| Alternate Scenario | If customer search is not valid or does not exist the app shows are this product doesn’t exist and not matched details the app notifies the customer a message this product doesn’t exist and then goes back or returns to step 1 of Main Success Scenario to enter again. | |

**UC #6:**

**Product delivery:**

|  |  |  |
| --- | --- | --- |
| Use-case Number | UC #6 | |
| Use-Case Name | Product delivery | |
| Priority | Medium | |
| Actor | Customer | |
| Description | This use case describes allows the customer to order self-defense training tools and Sanitary products by choosing from the list | |
| Precondition | UC #1,3,4 | |
| Post-condition | If the use case was successful, the customer can order and delivered self-defense tools by using Womanista. If not, the app state is unchanged. | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The customer selects the Self Defense and Sanitary products from the list 2. Enter address for delivery 3. Select payment option and clicks confirm order | 1. App shows the price and payment option. 2. The app shows Order Confirm successfully message to the customer and track order. 3. Use case Exit. |
| Alternate Scenario | If customer details are not valid or does not exist the app shows are not filled out and not matched details the app notifies the customer a message Order Canceled and then goes back or returns to step 1 of Main Success Scenario to enter again. | |

**UC #7:**

**View Women Protection Law:**

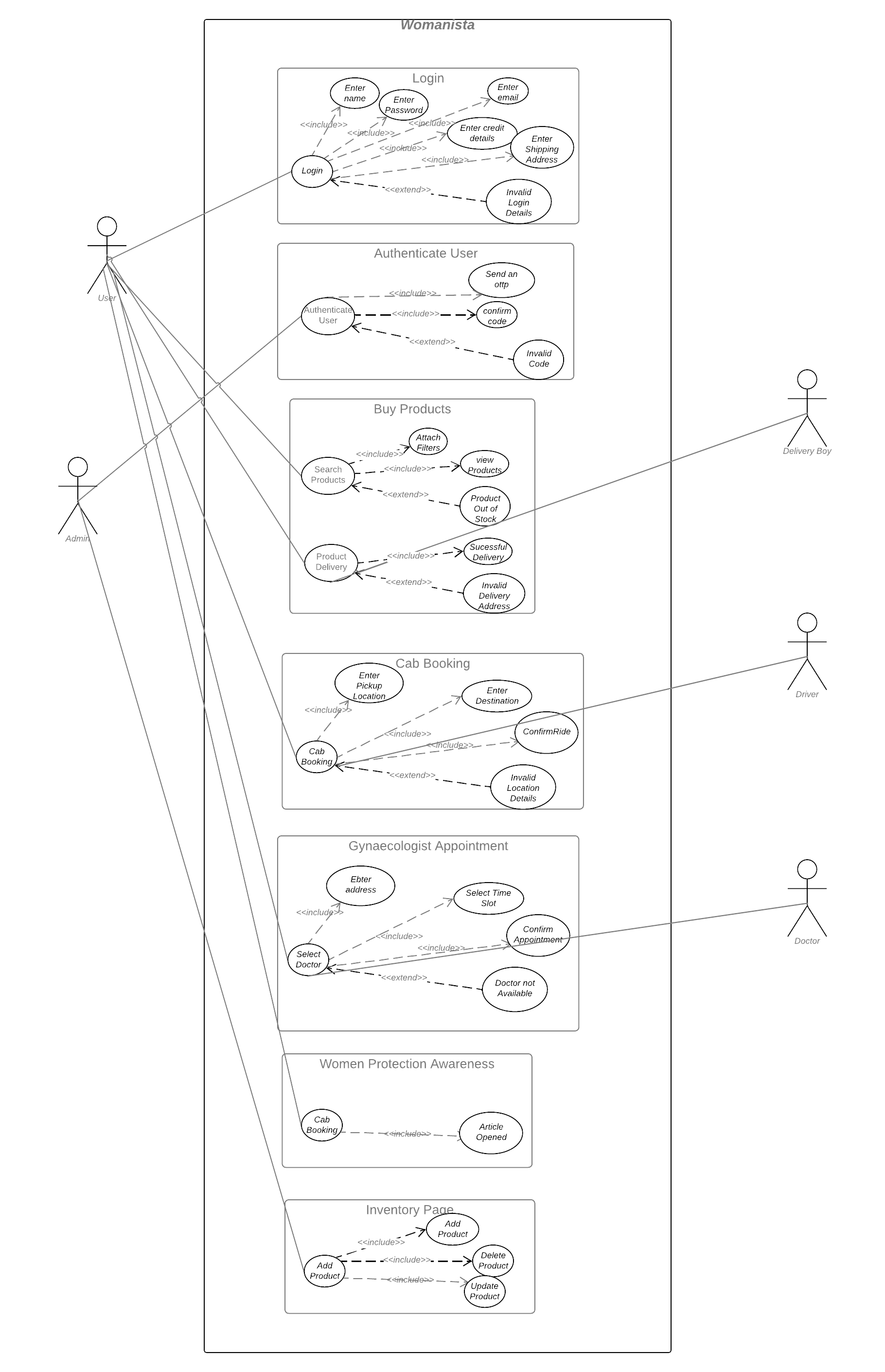
|  |  |  |
| --- | --- | --- |
| Use-case Number | UC #7 | |
| Use-Case Name | View Women Protection Law | |
| Priority | Medium | |
| Actor | Customer | |
| Description | This use case describes allows the customer to read Women Protection Laws | |
| Precondition | UC #1,3,4 | |
| Post-condition | If the use case was successful, the customer can now read Women Protection Laws by using Womanista. If not, the app state is unchanged. | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. The customer selects Women Protection Laws button from the app 2. Documents are shown | 1. App shows Documents are shown of Women Protection Laws 2. Use case Exit. |
| Alternate Scenario | If customer details are not valid or does not exist the app shows are not filled out and not matched details the app notifies the customer a message Not Women Please Authenticate then goes back or returns to step 1 of Main Success Scenario to enter again. | |

**UC #8:**

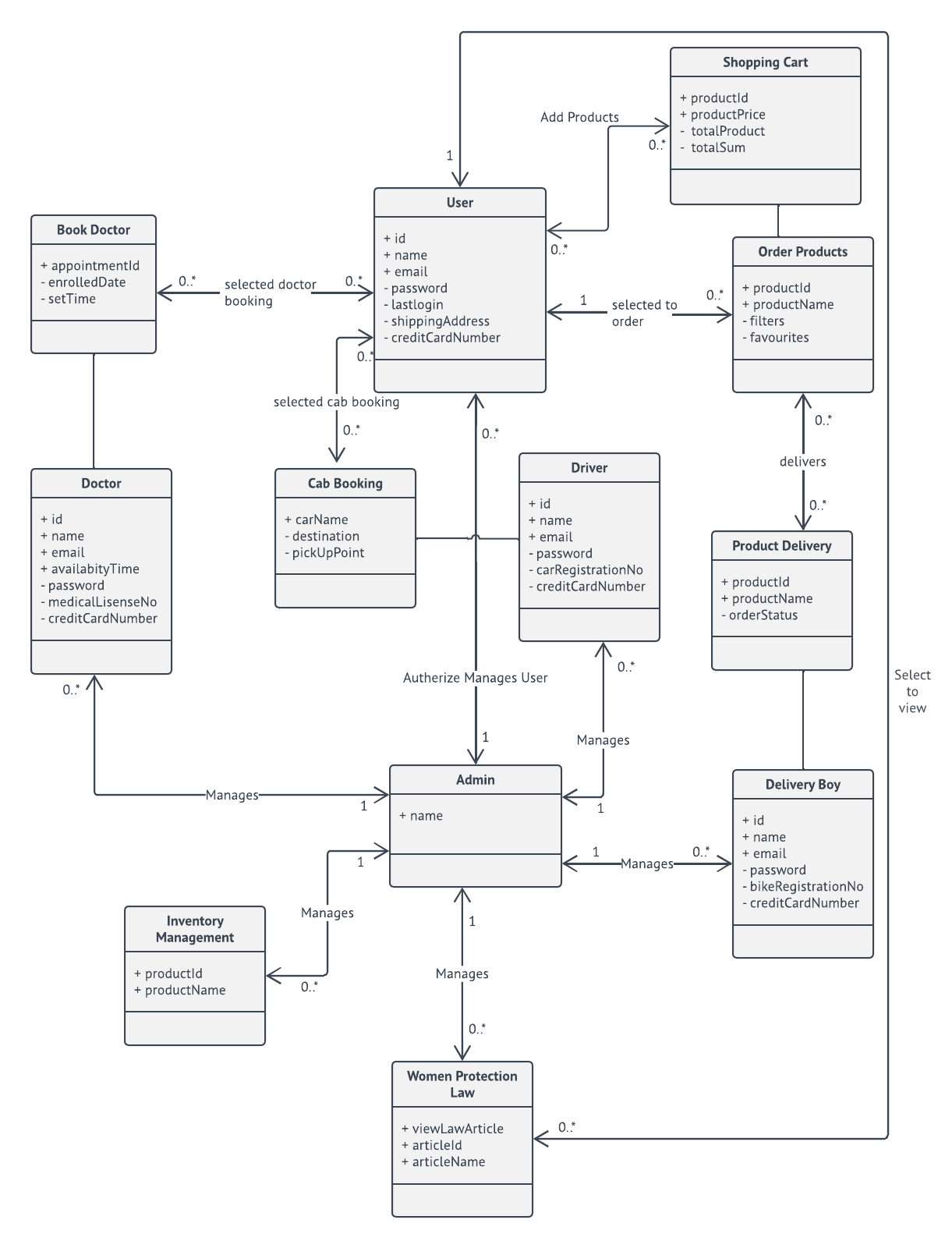
**Manage Inventory:**

|  |  |  |
| --- | --- | --- |
| Use-Case Number | UC #11 | |
| Use-Case Name | Manage Inventory | |
| Priority | Low | |
| Actor | Admin | |
| Description | This use case permits admin to update or modify product information in case when there is a need for editing | |
| Precondition | Need to Change information | |
| Post-condition | Successful Update Message | |
| Main Success Scenario | **User Action** | **App Response** |
| 1. Admin wants to update product.  2.Open the inventory page  3. Search by unique attribute which is given to admin during product.  5.The Admin update the information  6. Click on update button. | 4. The app displays the product information.  7. The app validates updated information and saves updated information in to database.  8. Exit use case. |
| Alternate Scenario | 4.1 If match is not found go back to Main Success Scenario 3.  7.1if the entered information is invalid the app back to Main Success Scenario 5 | |

## 1.2 Usecase Diagram (refined and updated)

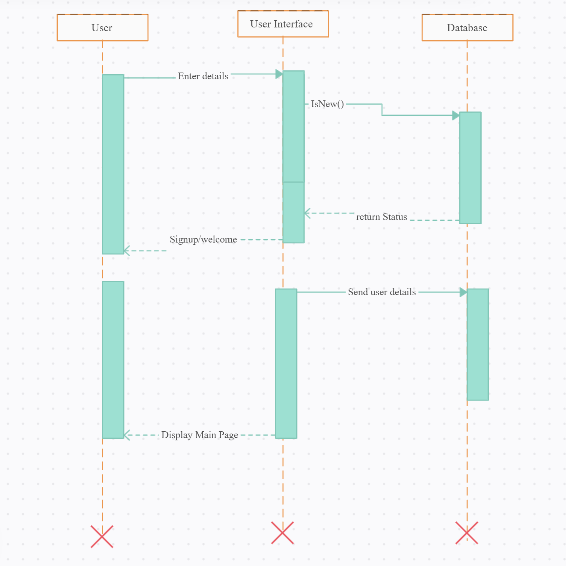


## 1.3 Domain Model

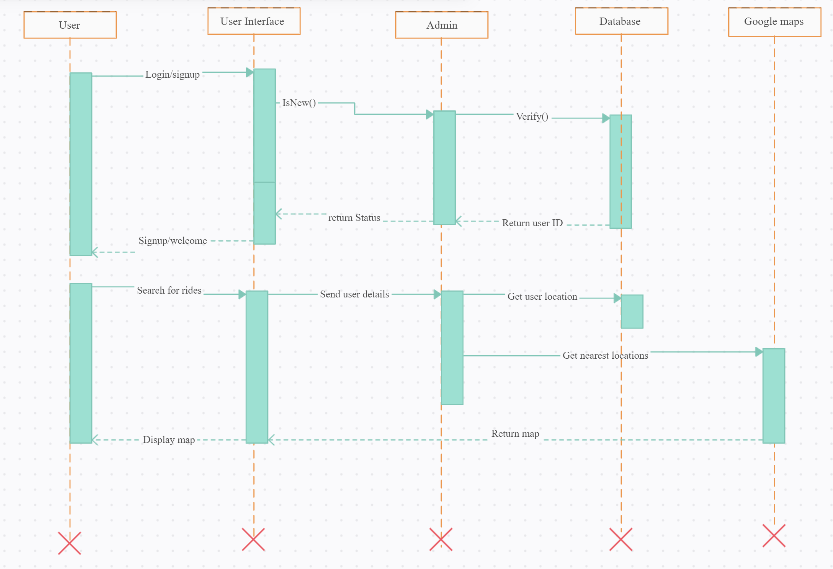


## 1.4 Sequence Diagram

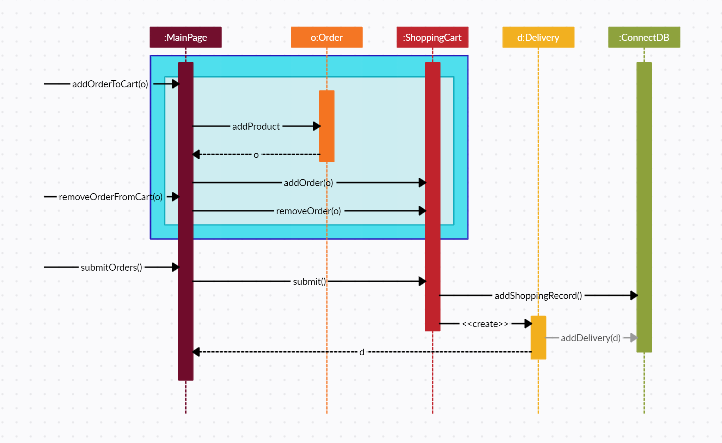
**LOGIN:**

****

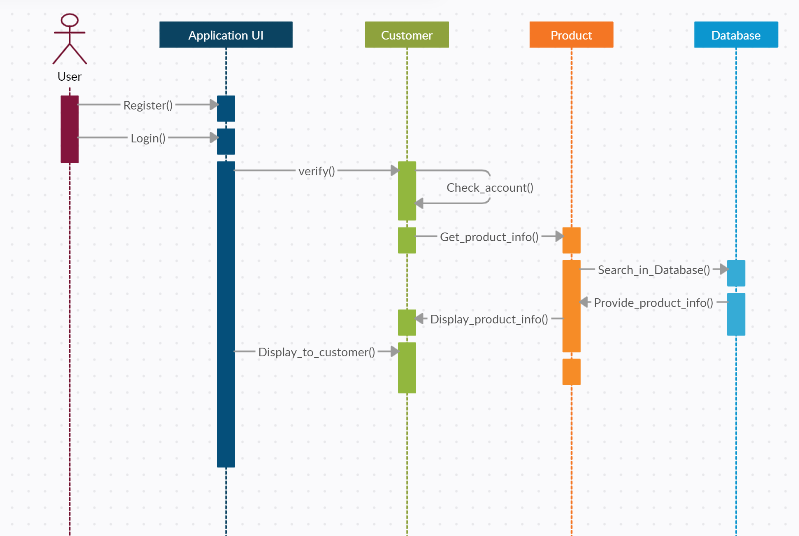
**GOOGLE MAPS:**

****

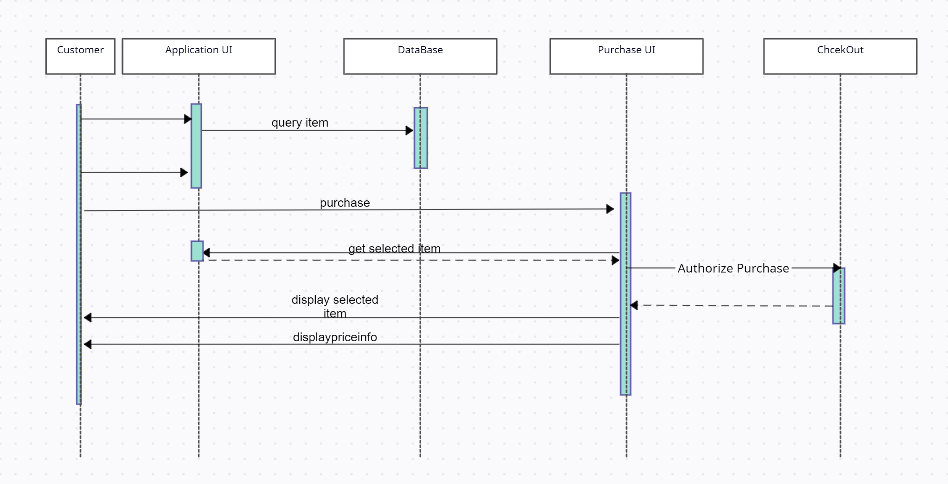
**ADD/REMOVE FROM CART:**

****

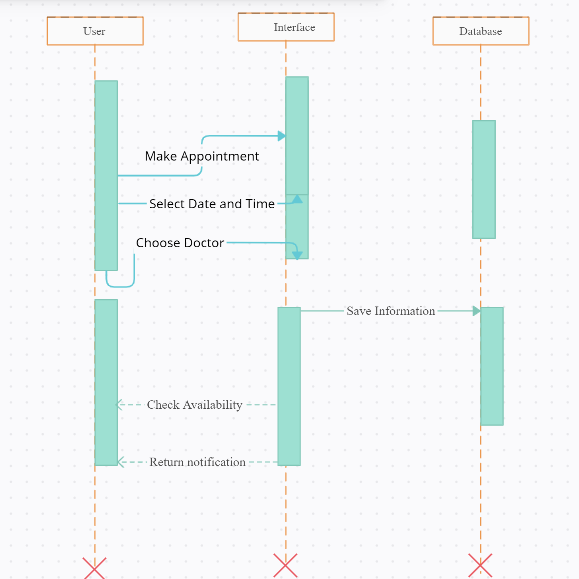
**SEARCH PRODUCTS:**

****

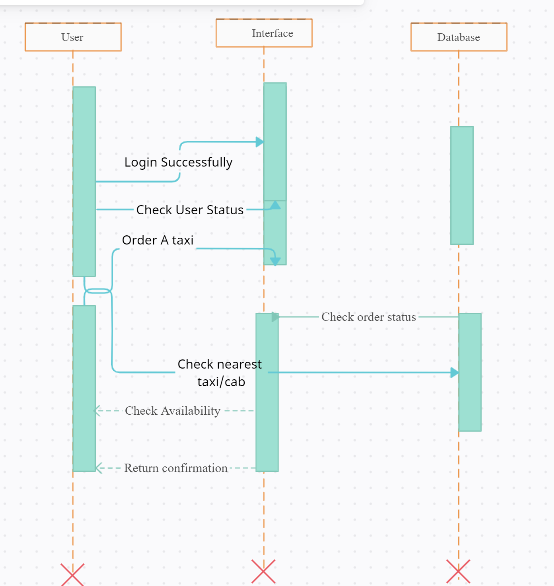
**PURCHASE PRODUCTS:**

****

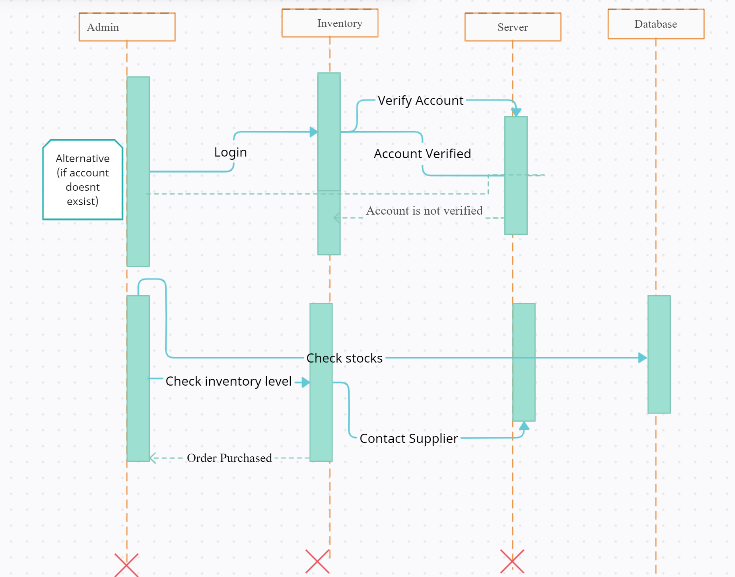
**DOCTOR APPOINTMENT:**

****

**CAB BOOKING:**

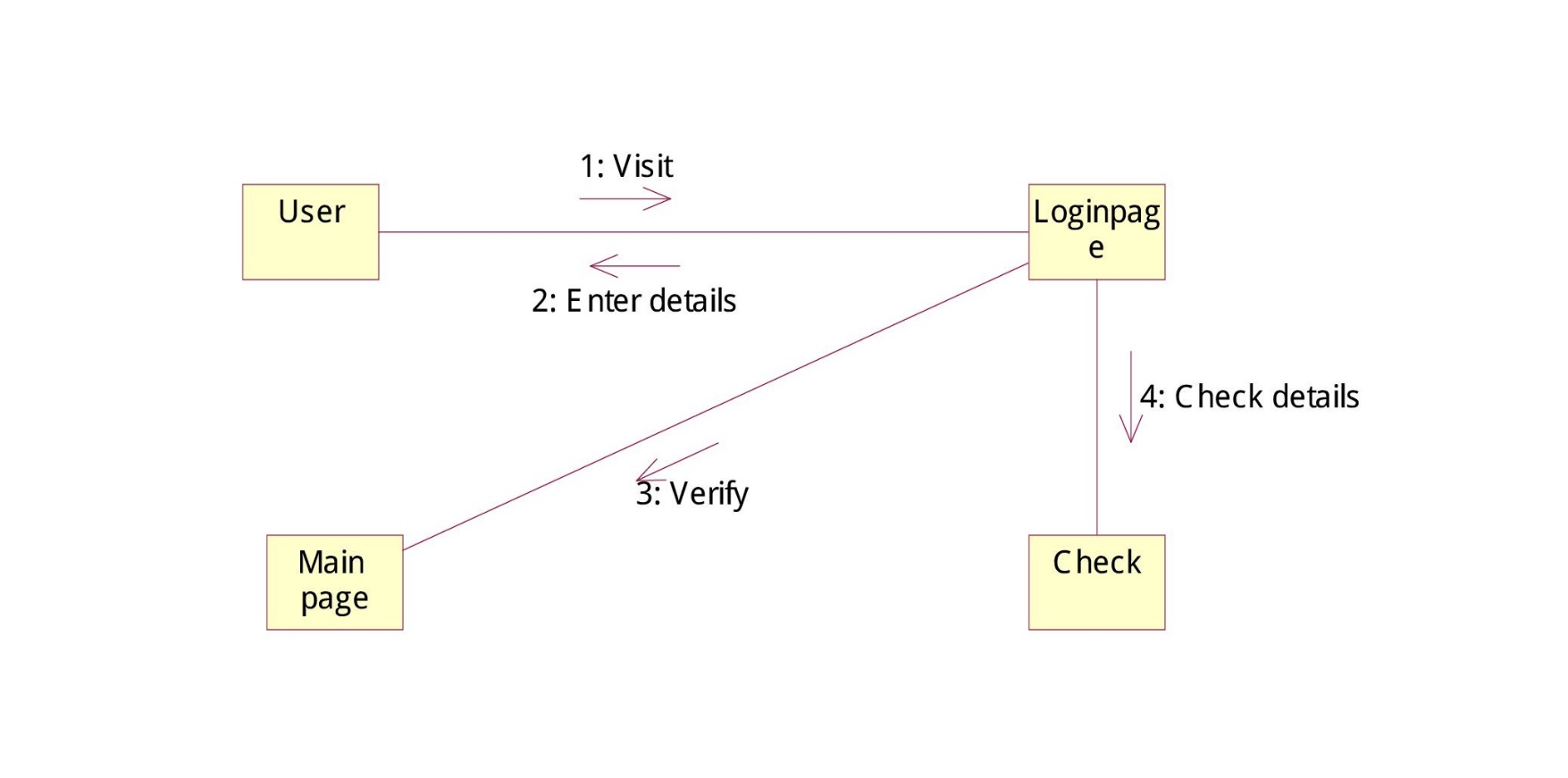
****

**INVENTORY:**

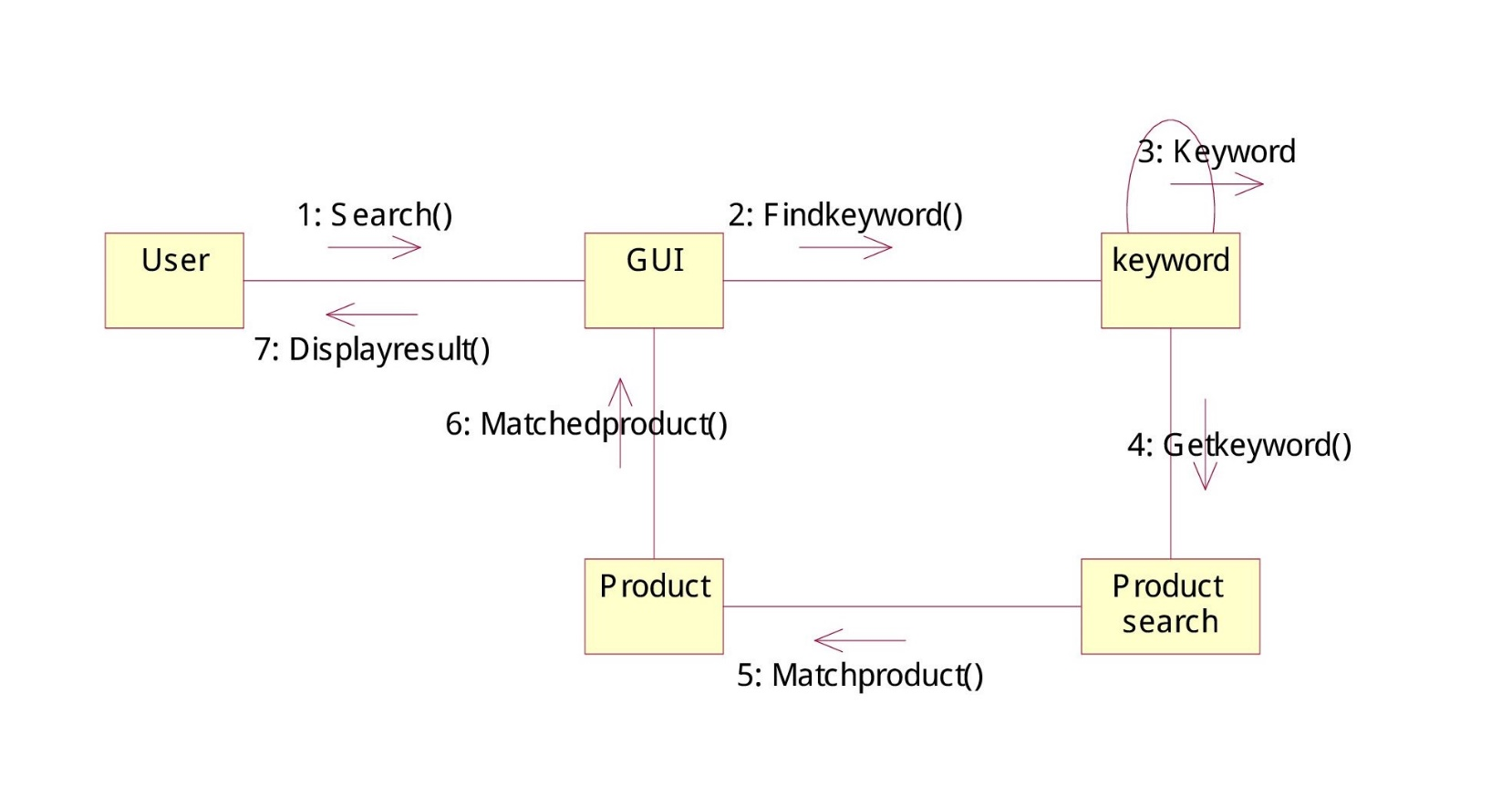
****

## 1.5 Collaboration Diagram

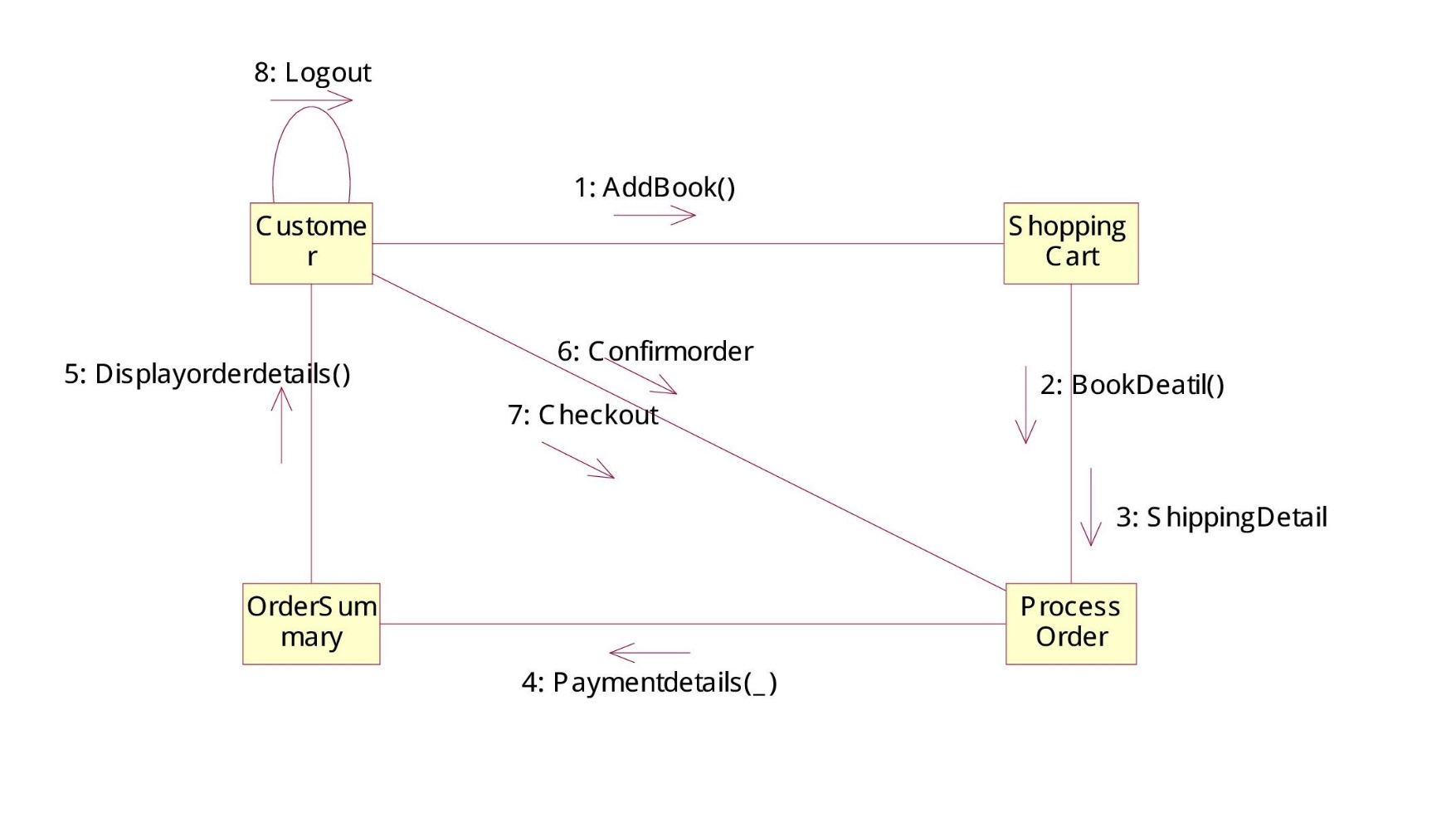
**LOGIN:**

****

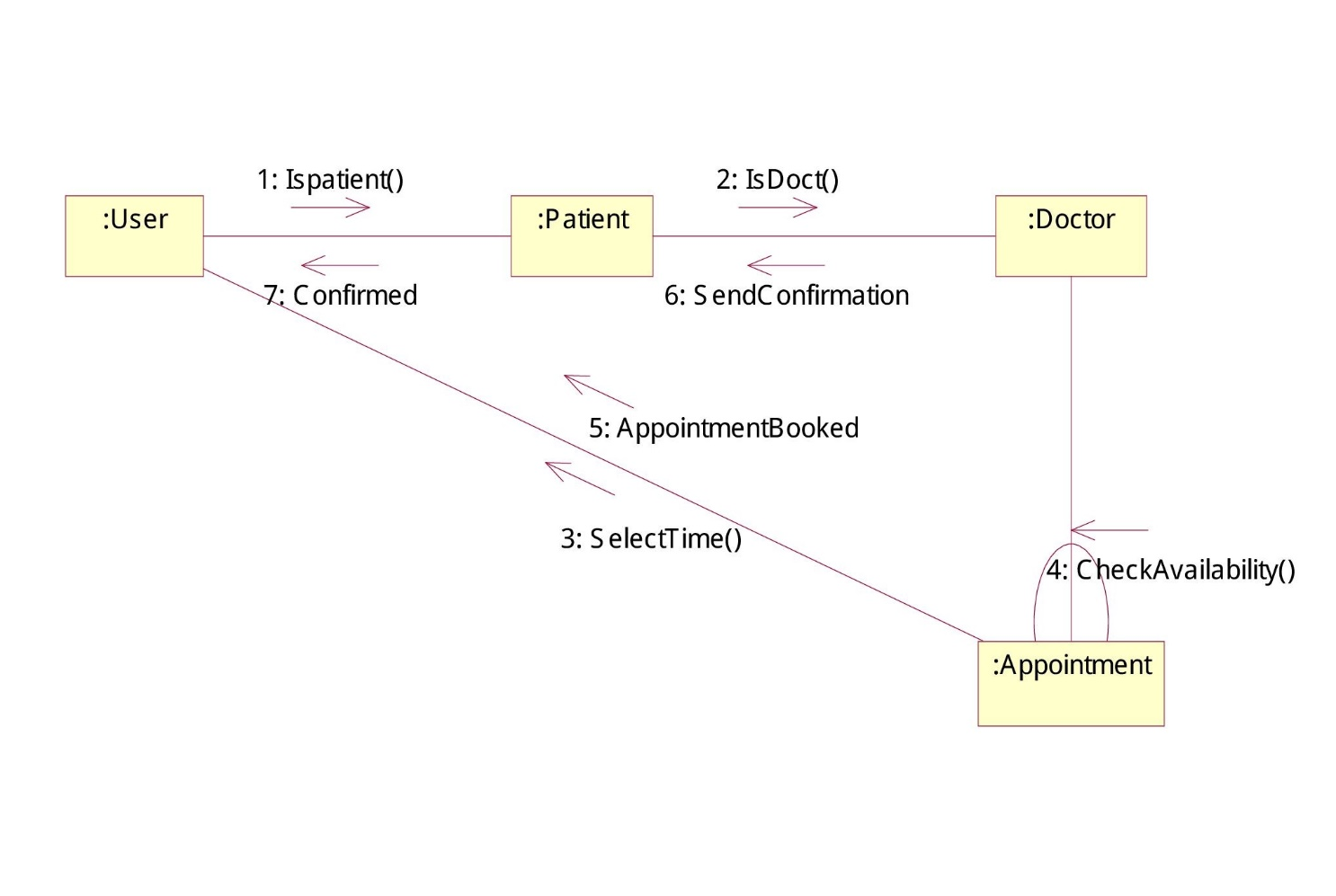
**SEARCH:**

****

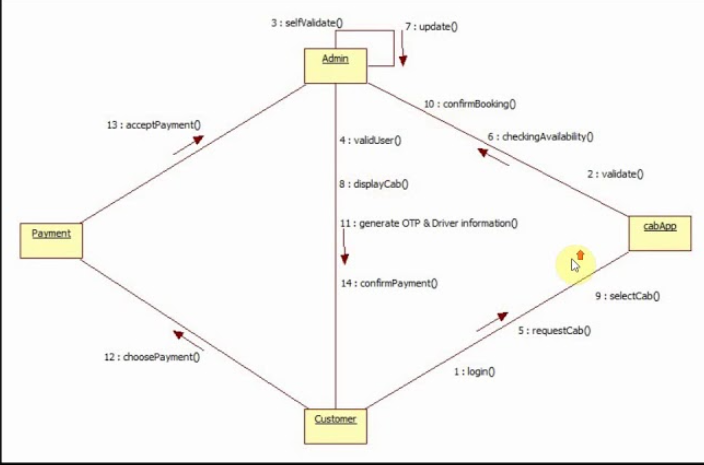
**PURCHASE:**

****

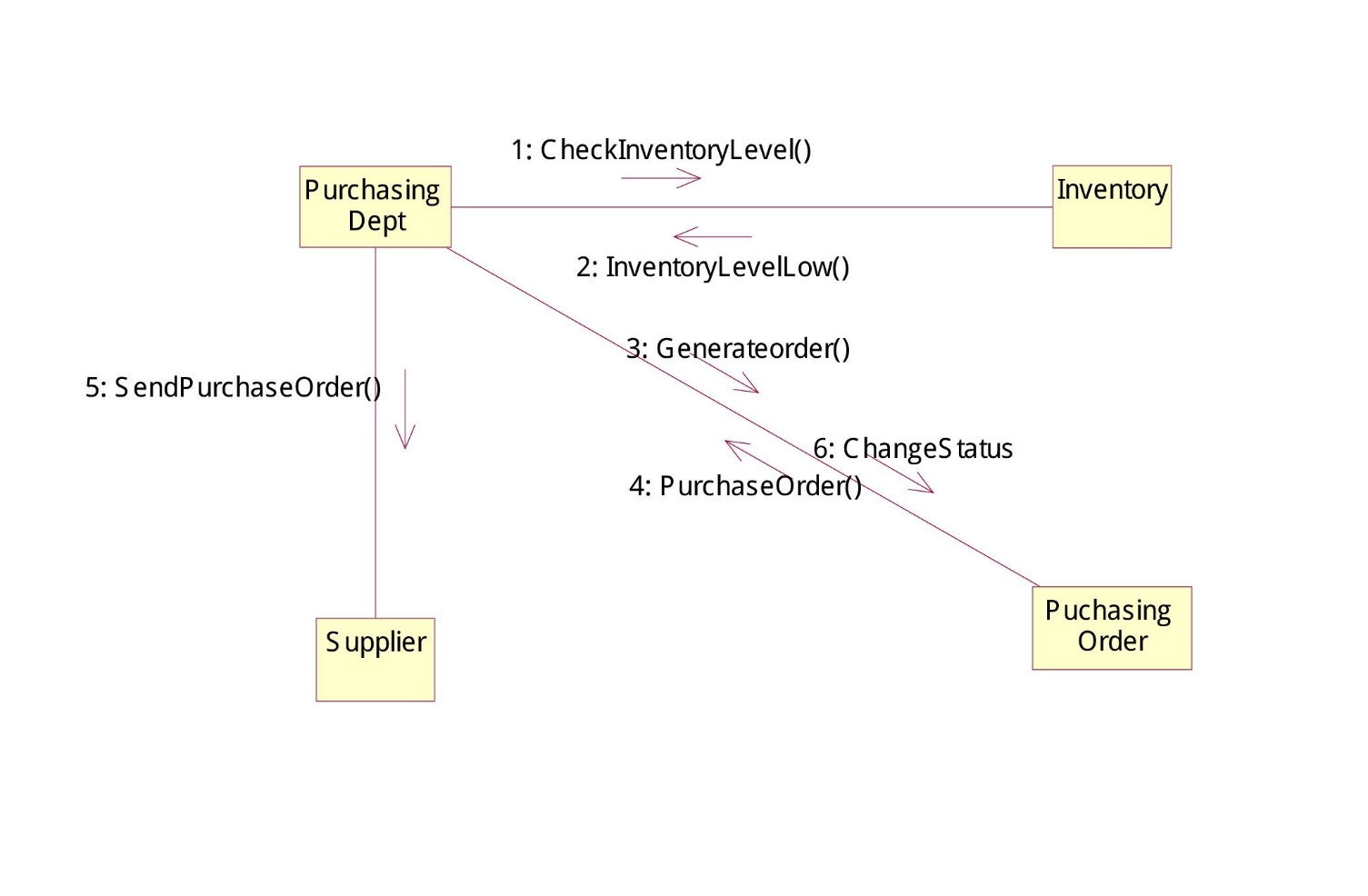
**DOCTOR APPOINTMENT:**

****

**CAB BOOKING:**

****

**INVENTORY:**

****

## 

## 1.6 Operation Contracts

**Contract Sections – EnterDetails()**

|  |  |
| --- | --- |
| **Operation:** | enterDetails (UserId: Id, Userpassword:string) |
| **Cross References:** | Use cases: Login |
| **Preconditions:** | There is a login operation underway. |
| **Post-conditions:** | The state of objects in the Domain Model after completion of the operation. The user is now logged into the App. |

**Contract Sections – SelectList()**

|  |  |
| --- | --- |
| **Operation:** | SelectList (SelectProduct: products) |
| **Cross References:** | Use cases: SearchProduct |
| **Preconditions:** | There is a Sale operation underway. |
| **Post-conditions:** | The state of objects in the Domain Model after completion of the operation. The user will select the items he wants to buy. |

**Contract Sections – MakeSale()**

|  |  |
| --- | --- |
| **Operation:** | MakeSale (ItemId:ItemId,quantity:Integar, UserAddress: string) |
| **Cross References:** | Use cases: ProductDelivery |
| **Preconditions:** | There is a delivery operation underway. |
| **Post-conditions:** | The state of objects in the Domain Model after completion of the operation. The user will enter address and products are delivered. |

**Contract Sections – SelectCab()**

|  |  |
| --- | --- |
| **Operation:** | SelectCab (UserPickup:Address) |
| **Cross References:** | Use cases: CabBooking |
| **Preconditions:** | There is a cab booking operation underway. |
| **Post-conditions:** | The state of objects in the Domain Model after completion of the operation. The user will select the booked the cab he wants to ride in. |

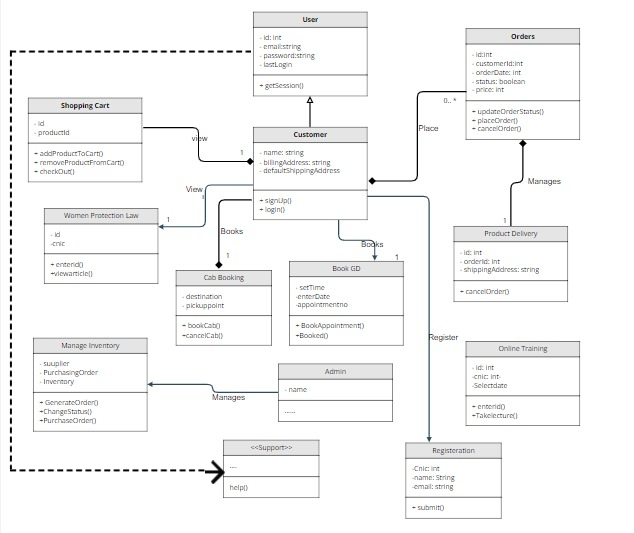
**Contract Sections – SelectLawArticles()**

|  |  |
| --- | --- |
| **Operation:** | SelectLawArticles (ViewArticle, EnterId:int, UserId:int) |
| **Cross References:** | Use cases: View Women Protection Law |
| **Preconditions:** | There is a Viewing Women Protection Law operation underway. |
| **Post-conditions:** | The state of objects in the Domain Model after completion of the operation. The user will read the Women Protection Law and get awareness |

**Contract Sections – Add/RemoveProducts()**

|  |  |
| --- | --- |
| **Operation:** | Add/RemoveProducts (Add/RemoveProducts: products) |
| **Cross References:** | Use cases: Manage Inventory |
| **Pre-conditions:** | There is a managing inventory operation underway. |
| **Post-conditions:** | The state of objects in the Domain Model after completion of the operation. The ADMIN will select the Add/Remove Products he wants to update in the inventory. |

## 1.7 Design Class Diagram

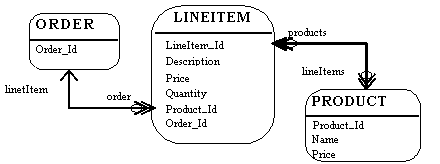


## 1.8 Data Model

The data model is a subset of the implementation model, which describes the logical and physical representation of persistent data in the system.

**The Relational Data Model**

The relational model is composed of entities and relations. An entity may be a physical table or a logical projection of several tables also known as a view. The figure below illustrates LINEITEM and PRODUCT tables and the various relationships between them.



A relational model has the following elements:

An entity has columns. A name and a type identify each column. In the figure above, the LINEITEM entity has the columns LineItem\_Id (the primary key), Description, Price, Quantity, Product\_Id and Order\_Id (the latter two are foreign keys that link the LINEITEM entity to the ORDER and PRODUCT entities).

An entity has records or rows. Each row represents a unique set of information, which typically represents an object's persistent data. Each entity has one or more primary keys. The primary keys uniquely identify each record (for example, Id is the primary key for LINEITEM table).

Support for relations is vendor specific. The example illustrates the logical model and the relation between the PRODUCT and LINEITEM tables. In the physical model relations are typically implemented using foreign key / primary key references. If one entity relates to another, it will contain columns, which are foreign keys. Foreign key columns contain data, which can relate specific records in the entity to the related entity.

Relations have multiplicity (also known as cardinality). Common cardinalities are one to one (1:1), one to many (1:m), many to one (m:1), and many to many (m:n). In the example, LINEITEM has a 1:1 relationship with PRODUCT and PRODUCT has a 0:m relationship with LINEITEM.

Example

A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.

1. Identify Entities

The entities in this system are Customer, Doctor, Delivery Boy, Driver and Product.

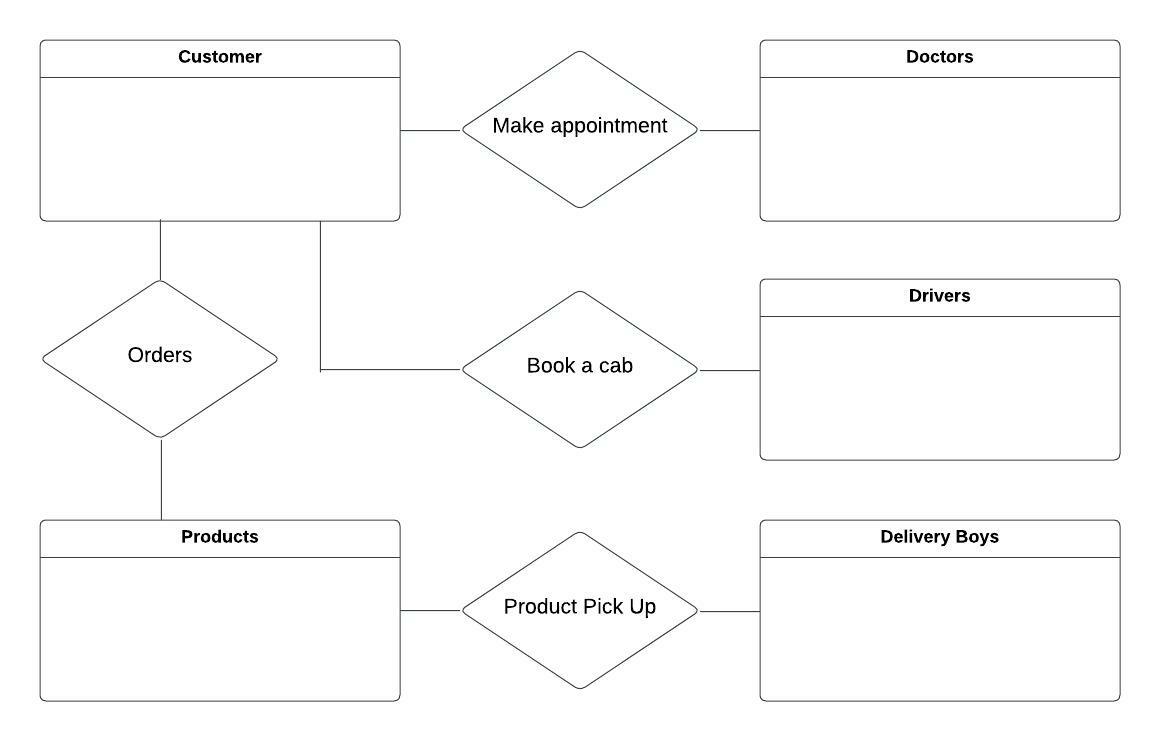
2. Find Relationships

We construct the following Entity Relationship Matrix:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Customer** | **Doctor** | **Delivery Boy** | **Product** | **Driver** |
| Customers |  | Make appointment |  | Orders | Book a cab |
| Doctors |  |  |  |  |  |
| Delivery Boys |  |  |  | Product Pick Up |  |
| Products |  |  |  |  |  |
| Drivers |  |  |  |  |  |

3. Draw Rough ERD

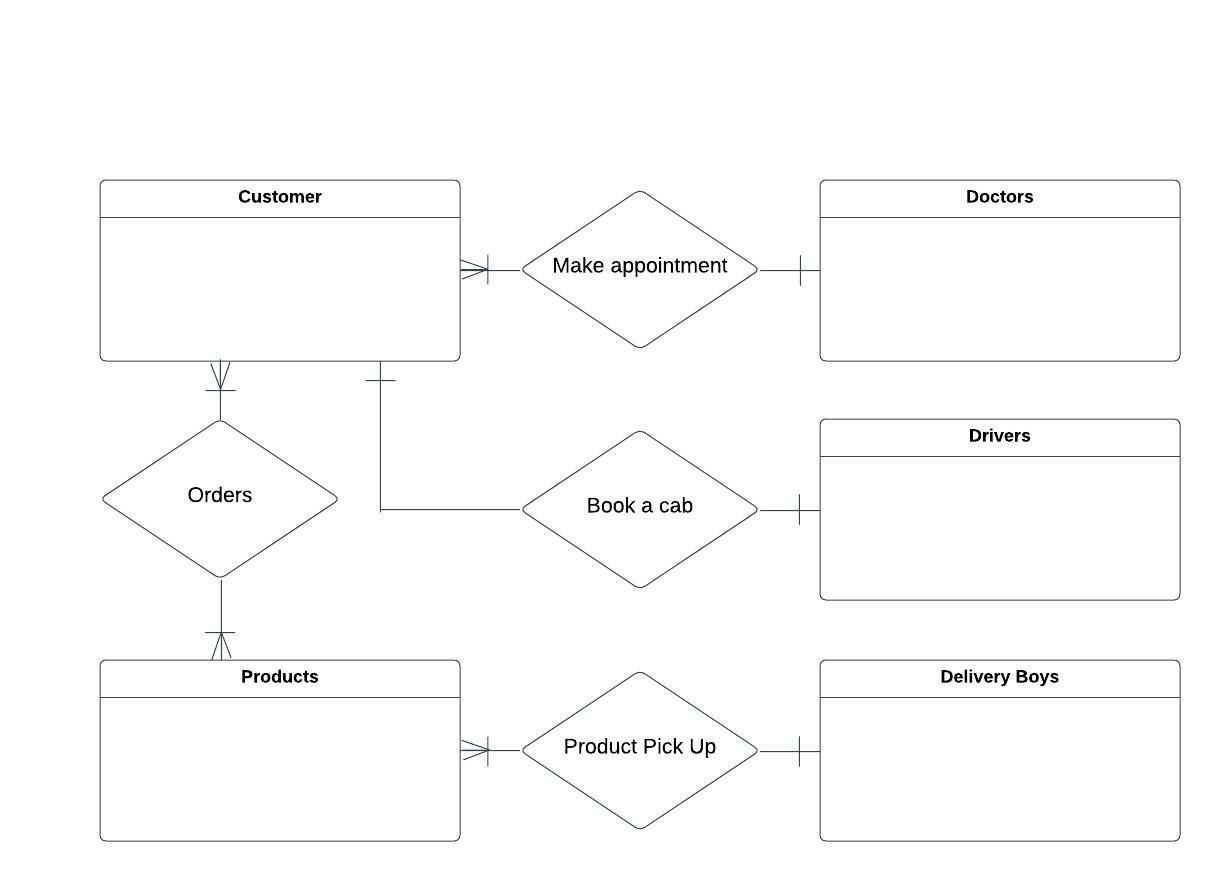
We connect the entities whenever a relationship is shown in the entity Relationship Matrix.



4. Fill in Cardinality

From the description of the problem we see that:

* Each department has exactly one supervisor.
* A supervisor is in charge of one and only one department.
* Each department is assigned at least one employee.
* Each employee works for at least one department.
* Each project has at least one employee working on it.
* An employee is assigned to 0 or more projects.



5. Define Primary Keys

The primary keys are Customer Id , Rider Id , Doctor Id, Product Id, Driver Id

6. Draw Key-Based ERD

There are two many-to-many relationships in the rough ERD above, between Department and Employee and between Employee and Project. Thus we need the associative entities Department-Employee and Employee-Project. The primary key for Department-Employee is the concatenated key Department Name and Employee Number. The primary key for Employee-Project is the concatenated key Employee Number and Project Number.

7. Identify Attributes

The only attributes indicated are the customer name , product name, product id, customer location, doctor name, driver name, rider name, driver cnic number

8. Map Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Entity** | **Attribute** | **Entity** |
| Customer Name | Customer | Doctor Name | Doctor |
| Product Name | Product | Driver Name | Driver |
| Product Id | Product | Rider Name | Rider |
| Location | Customer | CNIC Number | Driver |

9. Draw Fully Attributed ERD

