STOCK MAINTENANCE

Now a day's people should purchase things in stores. It is the System used to maintain the product details and stock.

So in this stock maintenance have the details about the product, purchase, sales and stock what are the stocks we had. The product details contain product code, Product name, Opening Stock and Prices. These details are maintained in database. In the purchasing function we must have the details about the store, quantity and also price.

The Sales Details contain Date, Customer name, Product code, Quantity and Prices. The Stock Details contain product id, opening stock, purchase stock, current stock, and sales. The stock details are maintaining the database and view the stock between two dates

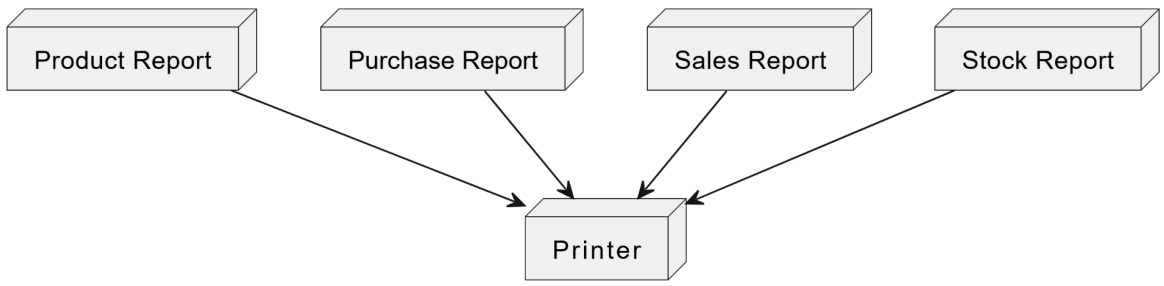
UML Diagrams:

1)DEPLOYMENT DIAGRAM:

CODE:

```
@startuml
node "Printer" {
}

node "Product Report" {
}
node "Purchase Report" {
}
node "Sales Report" {
}
node "Stock Report" {
}

"Product Report" --> "Printer"
"Purchase Report" --> "Printer"
"Sales Report" --> "Printer"
"Stock Report" --> "Printer"
@enduml
```

DIAGRAM:

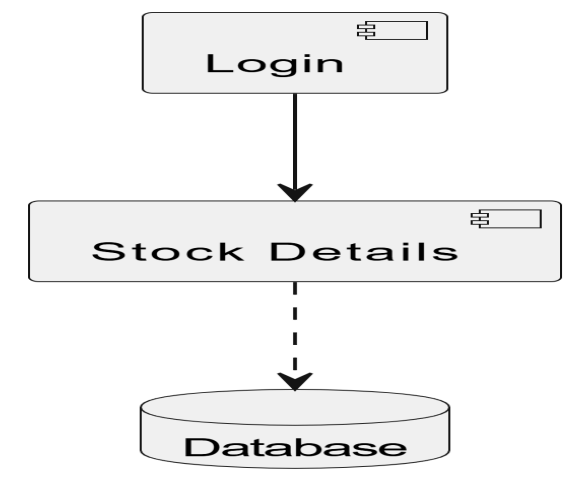


2) COMPONENT DIAGRAM:

CODE:

```
@startuml
component "Login" {
}

component "Stock Details" {
}

database "Database" {
}

"Login" --> "Stock Details"
"Stock Details" ..> "Database"
@enduml
```

DIAGRAM:



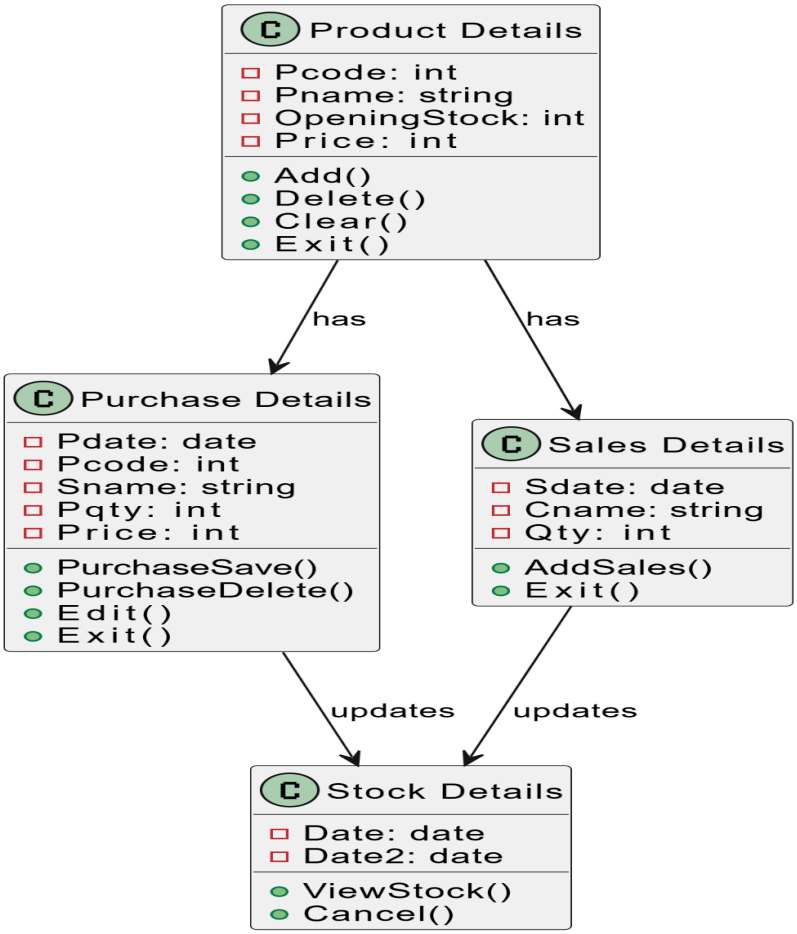
3)CLASS DIAGRAM:

CODE:

```
@startuml
class "Product Details" {
   - Pcode: int
   - Pname: string
   - OpeningStock: int
   - Price: int
   + Add()
   + Delete()
   + Clear()
   + Exit()
}

class "Purchase Details" {
   - Pdate: date
   - Pcode: int
   - Sname: string
   - Pqty: int
   - Price: int
   + PurchaseSave()
   + PurchaseDelete()
   + Edit()
   + Exit()
}

class "Sales Details" {
   - Sdate: date
   - Cname: string
   - Qty: int
   + AddSales()
   + Exit()
}

class "Stock Details" {
   - Date: date
   - Date2: date
   + ViewStock()
   + Cancel()
}

"Product Details" --> "Purchase Details"
"Product Details" --> "Sales Details"
```

"Purchase Details" --> "Stock Details"
"Sales Details" --> "Stock Details"
@enduml

DIAGRAM:



4) SEQUENCE DIAGRAM: (STOCK AND AVAILABILITY CHECK)

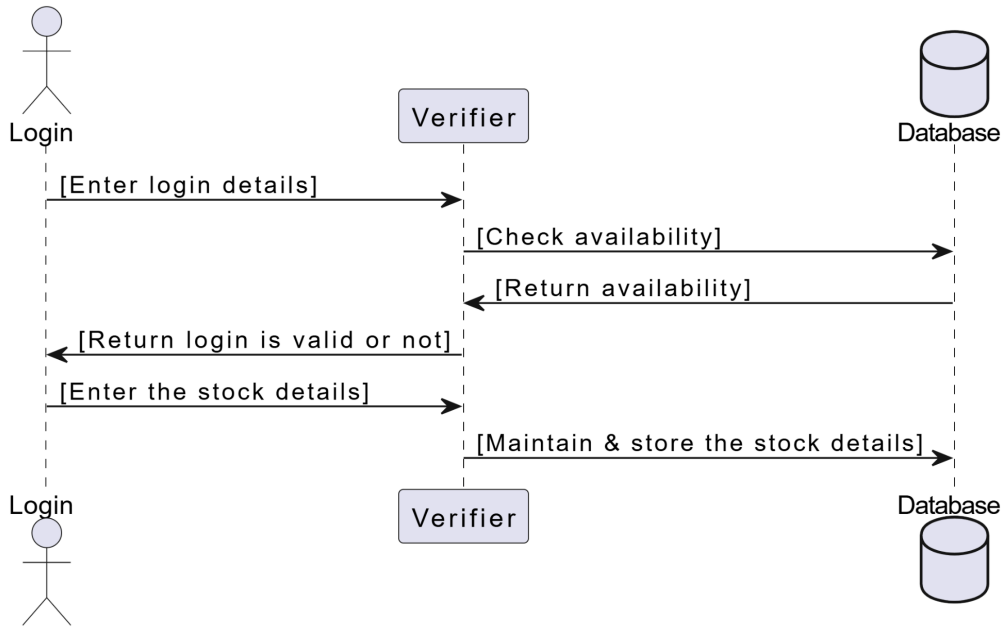
CODE:

@startuml
actor "Login" as User
participant "Verifier"
database "Database"

User -> Verifier: [Enter login details]
Verifier -> Database: [Check availability]
Database -> Verifier: [Return availability]

Verifier -> User: [Return login is valid or not]
User -> Verifier: [Enter the stock details]
Verifier -> Database: [Maintain & store the stock details]
@enduml

DIAGRAM:



5) USE CASE DIAGRAM:

CODE:
@startuml
actor "Administrator"
actor "Supplier"
actor "Customer"

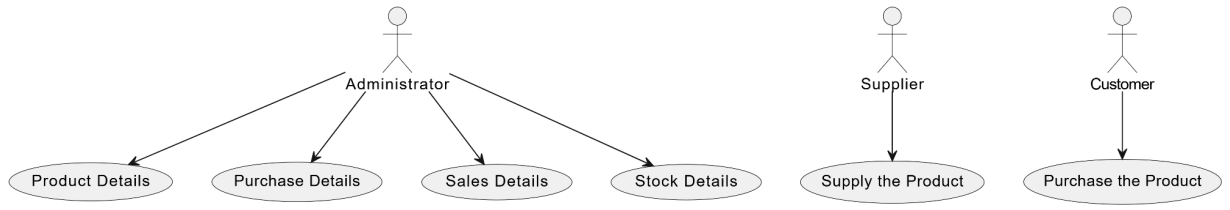
usecase "Product Details"
usecase "Purchase Details"
usecase "Sales Details"
usecase "Stock Details"
usecase "Purchase the Product"
usecase "Supply the Product"

"Administrator" --> "Product Details"
"Administrator" --> "Purchase Details"
"Administrator" --> "Sales Details"

"Administrator" --> "Stock Details"

"Supplier" --> "Supply the Product"
"Customer" --> "Purchase the Product"
@enduml

DIAGRAM:



6) State diagram:

Code:

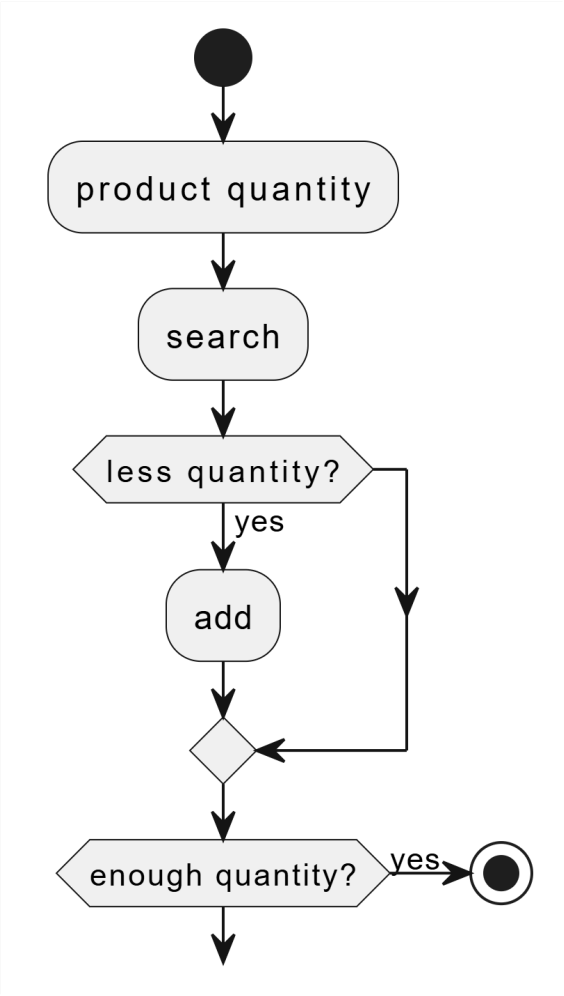
@startuml
start
:product quantity;
:search;

if (less quantity?) then (yes)
  :add;
endif

if (enough quantity?) then (yes)
  stop
endif

@enduml

DIAGRAM:



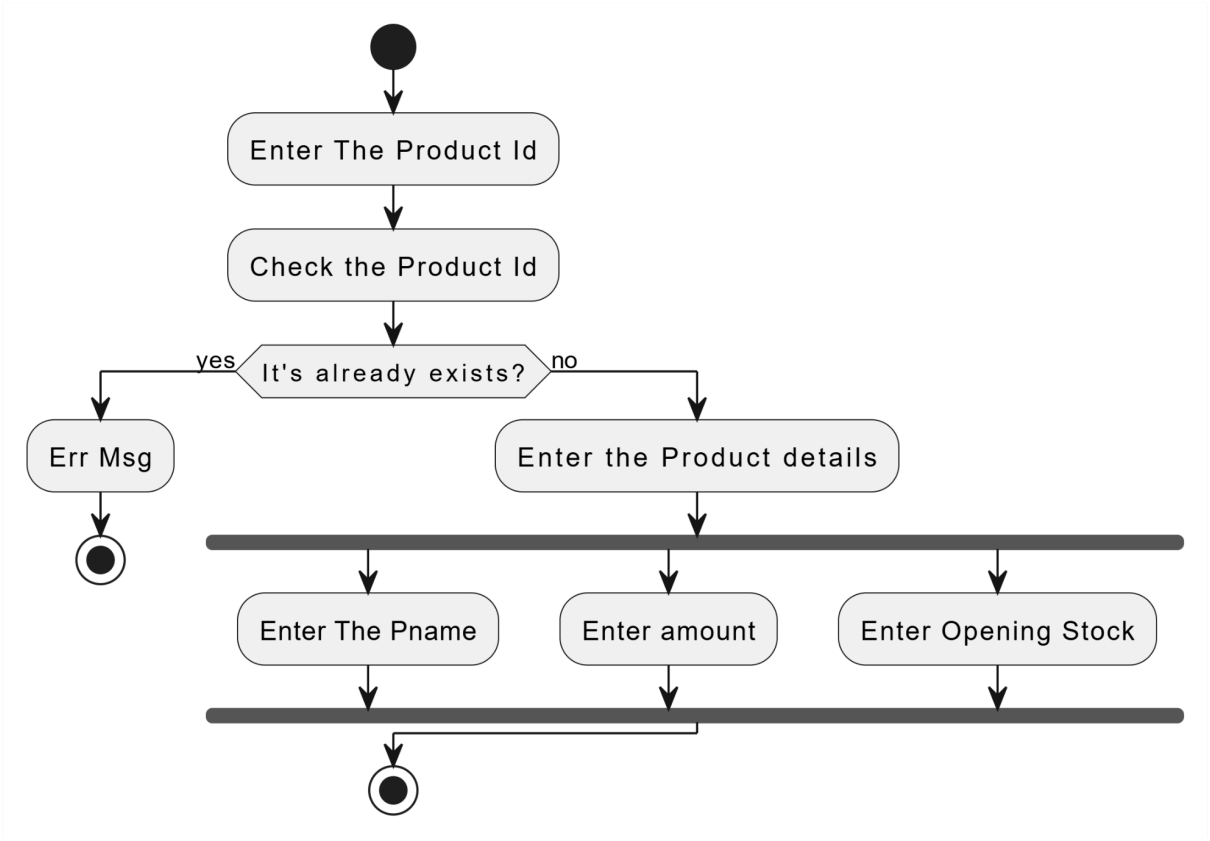
7) STATE DIAGRAM:

CODE:

```
@startuml
start
:Enter The Product Id;
:Check the Product Id;
if (It's already exists?) then (yes)
   :Err Msg;
   stop
else (no)
   :Enter the Product details;
   fork
```

```
        :Enter The Pname;
    fork again
        :Enter amount;
    fork again
        :Enter Opening Stock;
    end fork
endif
stop
@enduml
```

DIAGRAM:



8) SEQUENCE DIAGRAM:

CODE:

```
@startuml
participant "Login"
participant "Verifier"
database "Database"
```

"Login" -> "Verifier": [Enter the login details]
"Verifier" -> "Database": [Check availability]
"Database" -> "Verifier": [Return availability]
"Verifier" -> "Login": [Return login is valid or not]
"Login" -> "Verifier": [Enter the stock details]
"Verifier" -> "Database": [Maintain & store the stock details]
@enduml

DIAGRAM