EXPERIMENT-4

C.KRISHNA BALAJI

BU22CSEN0101063

Q) UML DIAGRAMS OF ONLINE BANKING SERVICES

USE CASE DIAGRAM

```
@startuml
left to right direction
actor Customer
actor BankEmployee
rectangle "Online Banking System" {
  usecase "Register Account" as UC1
  usecase "Login" as UC2
  usecase "Check Account Balance" as UC3
  usecase "Transfer Funds" as UC4
  usecase "Pay Bills" as UC5
  usecase "View Transaction History" as UC6
  usecase "Apply for Loan" as UC7
  usecase "Manage Beneficiaries" as UC8
  usecase "Logout" as UC9
  usecase "Approve Loan Application" as UC10
}
Customer -- UC1
Customer -- UC2
Customer -- UC3
```

Customer -- UC4

Customer -- UC5

Customer -- UC6

Customer -- UC7

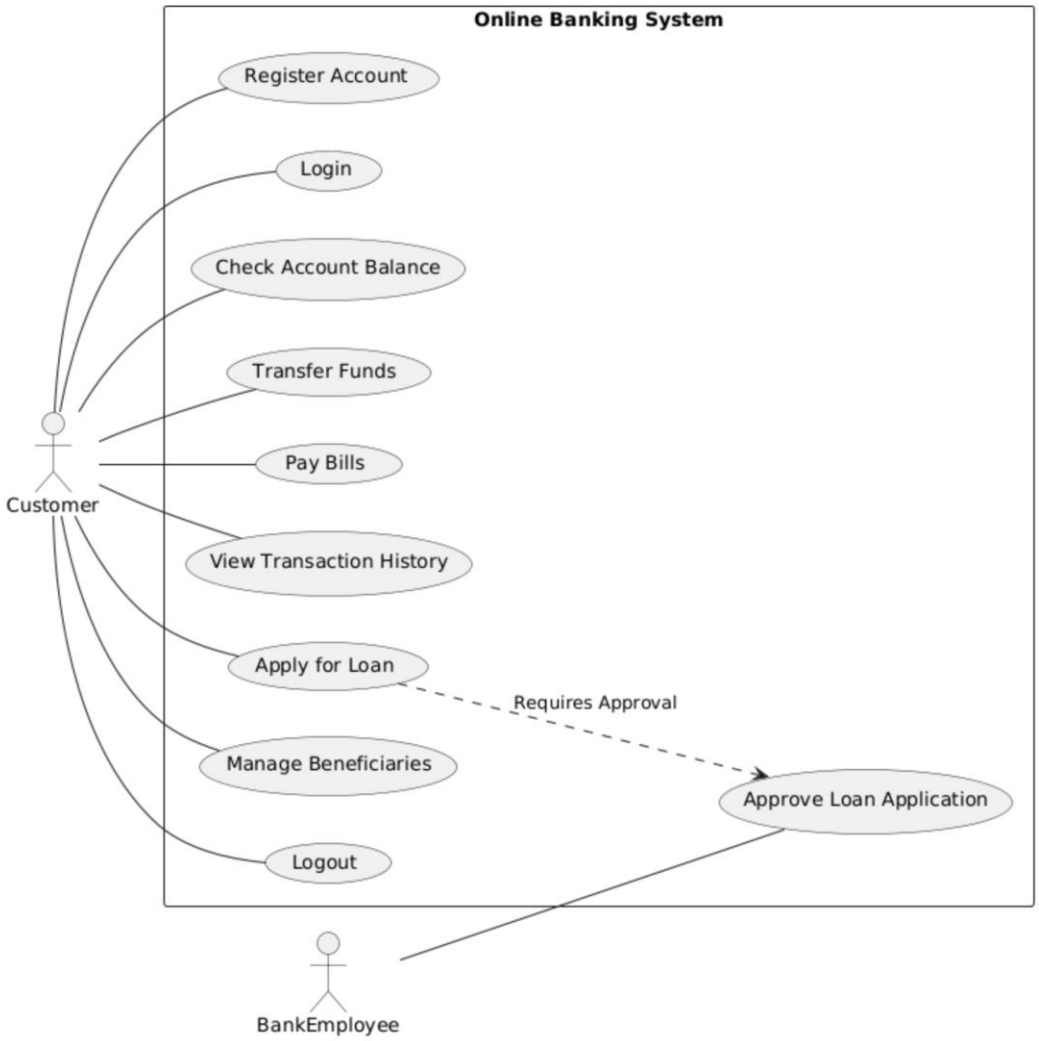
Customer -- UC8

Customer -- UC9

BankEmployee -- UC10

UC7 ..> UC10 : "Requires Approval"

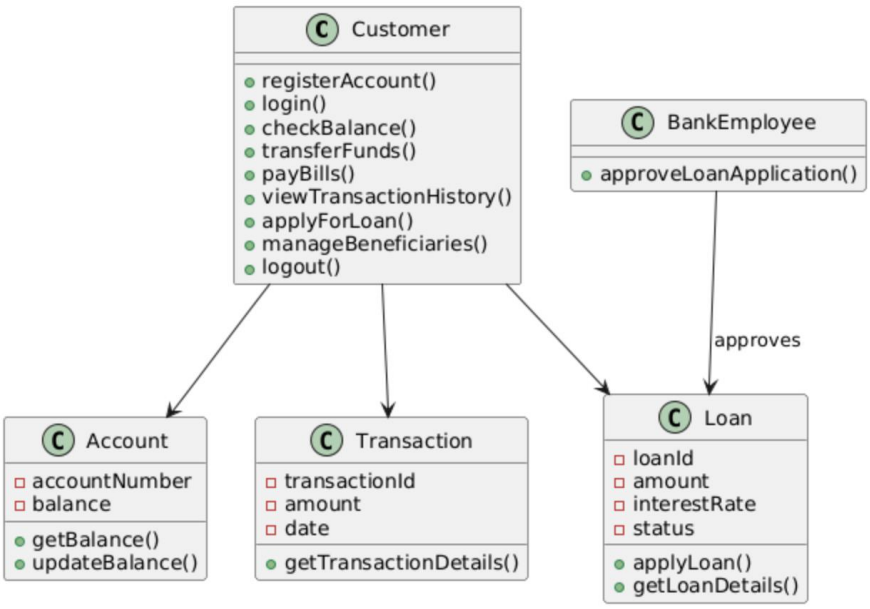
@enduml



CLASS DIAGRAM

```
@startuml
 class Customer {
   +registerAccount()
   +login()
   +checkBalance()
   +transferFunds()
   +payBills()
   +viewTransactionHistory()
   +applyForLoan()
   +manageBeneficiaries()
   +logout()
}
class BankEmployee {
   +approveLoanApplication()
}
class Account {
   -accountNumber
   -balance
   +getBalance()
   +updateBalance()
}
class Transaction {
   -transactionId
   -amount
```

```
    -date
    +getTransactionDetails()
}
class Loan {
    -loanId
    -amount
    -interestRate
    -status
    +applyLoan()
    +getLoanDetails()
}
Customer --> Account
Customer --> Transaction
Customer --> Loan
BankEmployee --> Loan : "approves"
 @enduml
```



SEQUENCE DIAGRAM

@startuml

actor Customer

participant "Online Banking System" as OBS

participant Account

participant Transaction

participant Loan

participant BankEmployee

Customer -> OBS: login()

OBS -> Account: verifyCredentials()

Account -> OBS: authenticationSuccess()

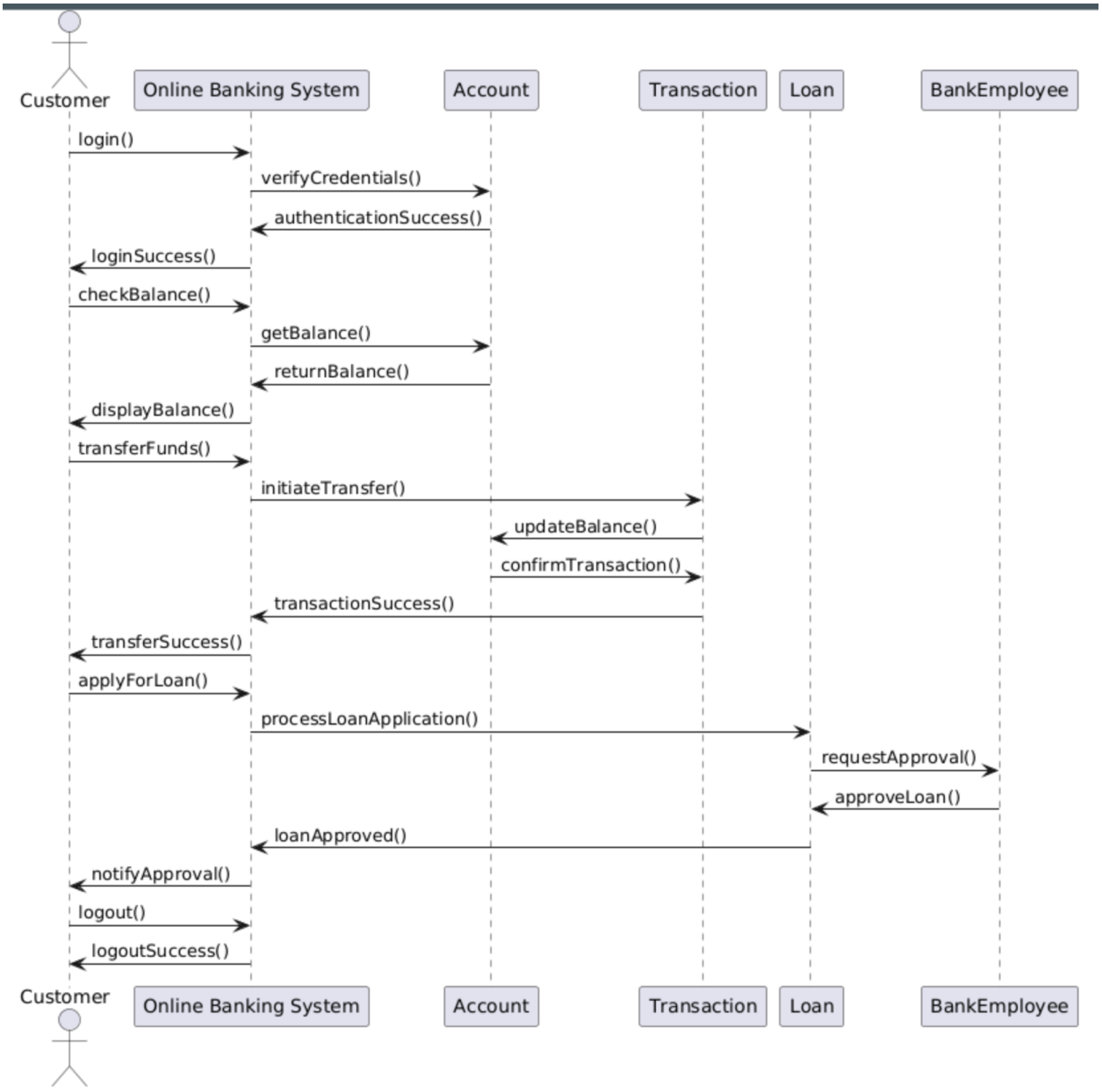
OBS -> Customer: loginSuccess()

Customer -> OBS: checkBalance()

OBS -> Account: getBalance()

Account -> OBS: returnBalance()

```
OBS -> Customer: displayBalance()

Customer -> OBS: transferFunds()

OBS -> Transaction: initiateTransfer()

Transaction -> Account: updateBalance()

Account -> Transaction: confirmTransaction()

Transaction -> OBS: transactionSuccess()

OBS -> Customer: transferSuccess()

Customer -> OBS: applyForLoan()

OBS -> Loan: processLoanApplication()

Loan -> BankEmployee: requestApproval()

BankEmployee -> Loan: approveLoan()

Loan -> OBS: loanApproved()

OBS -> Customer: notifyApproval()

Customer -> OBS: logout()

OBS -> Customer: logoutSuccess()

 @enduml
```



COLLABORATION DIAGRAM

@startuml

title Collaboration Diagram for Online Banking System

' Define objects in a vertical order

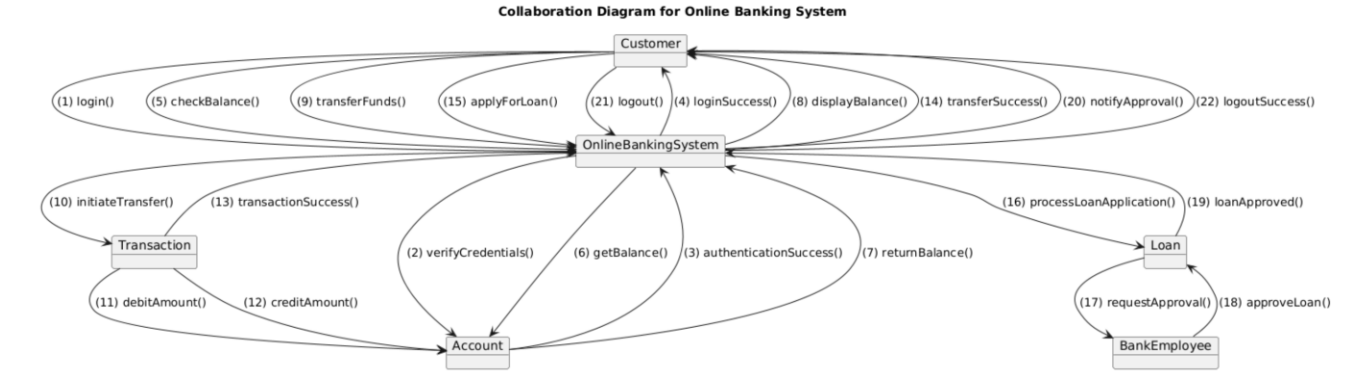
object Customer

object OnlineBankingSystem

```
object Account
object Transaction
object Loan
object BankEmployee
' Use top-to-bottom alignment for vertical layout
Customer -down-> OnlineBankingSystem : (1) login()
OnlineBankingSystem -down-> Account : (2) verifyCredentials()
Account -down-> OnlineBankingSystem : (3) authenticationSuccess()
OnlineBankingSystem -down-> Customer : (4) loginSuccess()
Customer -down-> OnlineBankingSystem : (5) checkBalance()
OnlineBankingSystem -down-> Account : (6) getBalance()
Account -down-> OnlineBankingSystem : (7) returnBalance()
OnlineBankingSystem -down-> Customer : (8) displayBalance()
Customer -down-> OnlineBankingSystem : (9) transferFunds()
OnlineBankingSystem -down-> Transaction : (10) initiateTransfer()
Transaction -down-> Account : (11) debitAmount()
Transaction -down-> Account : (12) creditAmount()
Transaction -down-> OnlineBankingSystem : (13) transactionSuccess()
OnlineBankingSystem -down-> Customer : (14) transferSuccess()
Customer -down-> OnlineBankingSystem : (15) applyForLoan()
OnlineBankingSystem -down-> Loan : (16) processLoanApplication()
Loan -down-> BankEmployee : (17) requestApproval()
BankEmployee -down-> Loan : (18) approveLoan()
Loan -down-> OnlineBankingSystem : (19) loanApproved()
OnlineBankingSystem -down-> Customer : (20) notifyApproval()
Customer -down-> OnlineBankingSystem : (21) logout()
```

OnlineBankingSystem -down-> Customer : (22) logoutSuccess()

@enduml



STATE CHART DIAGRAM

@startuml

[*] --> Idle

Idle --> LoggingIn : Enter Credentials

LoggingIn --> LoggedIn : Authentication Successful

LoggingIn --> Idle : Authentication Failed

LoggedIn --> CheckingBalance : Check Balance

CheckingBalance --> LoggedIn : Balance Displayed

LoggedIn --> TransferringFunds : Transfer Funds

TransferringFunds --> TransactionCompleted : Transfer Successful

TransferringFunds --> LoggedIn : Transfer Failed

LoggedIn --> ApplyingForLoan : Apply for Loan

ApplyingForLoan --> LoanApproved : Loan Approved

ApplyingForLoan --> LoanRejected : Loan Rejected

LoanApproved --> LoggedIn

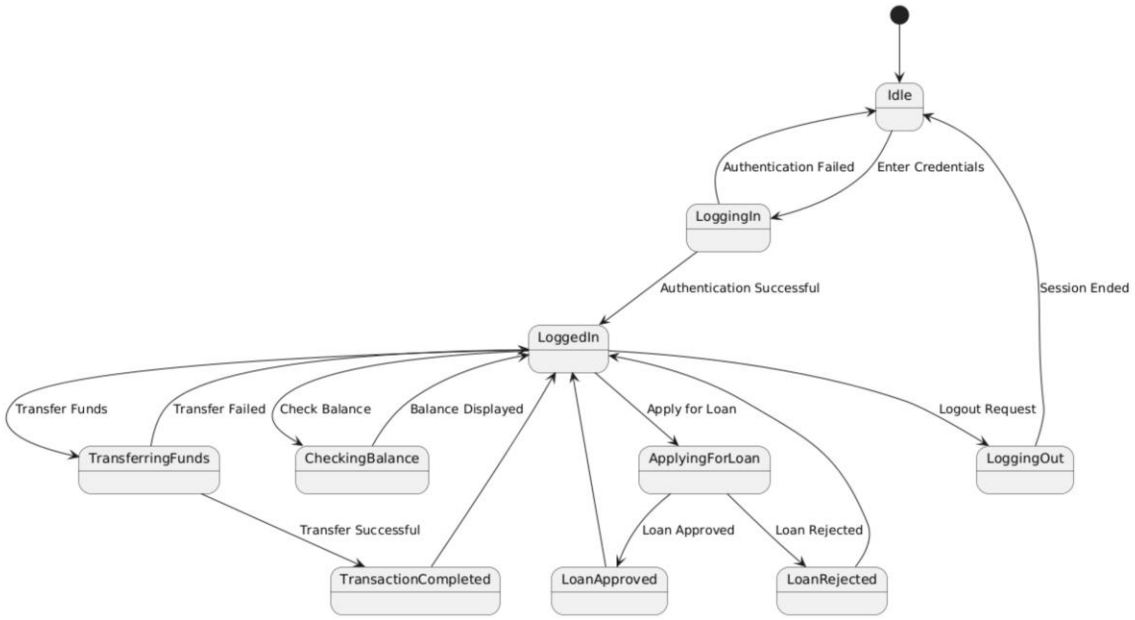
LoanRejected --> LoggedIn

TransactionCompleted --> LoggedIn

LoggedIn --> LoggingOut : Logout Request

LoggingOut --> Idle : Session Ended

@enduml



ACTIVITY DIAGRAM

@startuml

start

:User Opens Online Banking Portal;

:Enter Credentials;

if (Are Credentials Valid?) then (Yes)

```
:Display Dashboard;
repeat
  :Choose an Action;
  switch (Selected Action)
    case (Check Balance)
      :Retrieve Account Balance;
      :Display Balance;
    case (Transfer Funds)
      :Enter Transfer Details;
      if (Is Transfer Valid?) then (Yes)
        :Process Transfer;
        :Transfer Successful;
      else (No)
        :Show Error Message;
      endif
    case (Apply for Loan)
      :Submit Loan Application;
      if (Loan Approved?) then (Yes)
        :Approve Loan;
        :Notify User;
      else (No)
        :Reject Loan;
        :Notify User;
      endif
  endswitch
repeat while (User Wants Another Action?)
```

:Logout;

:End Session;

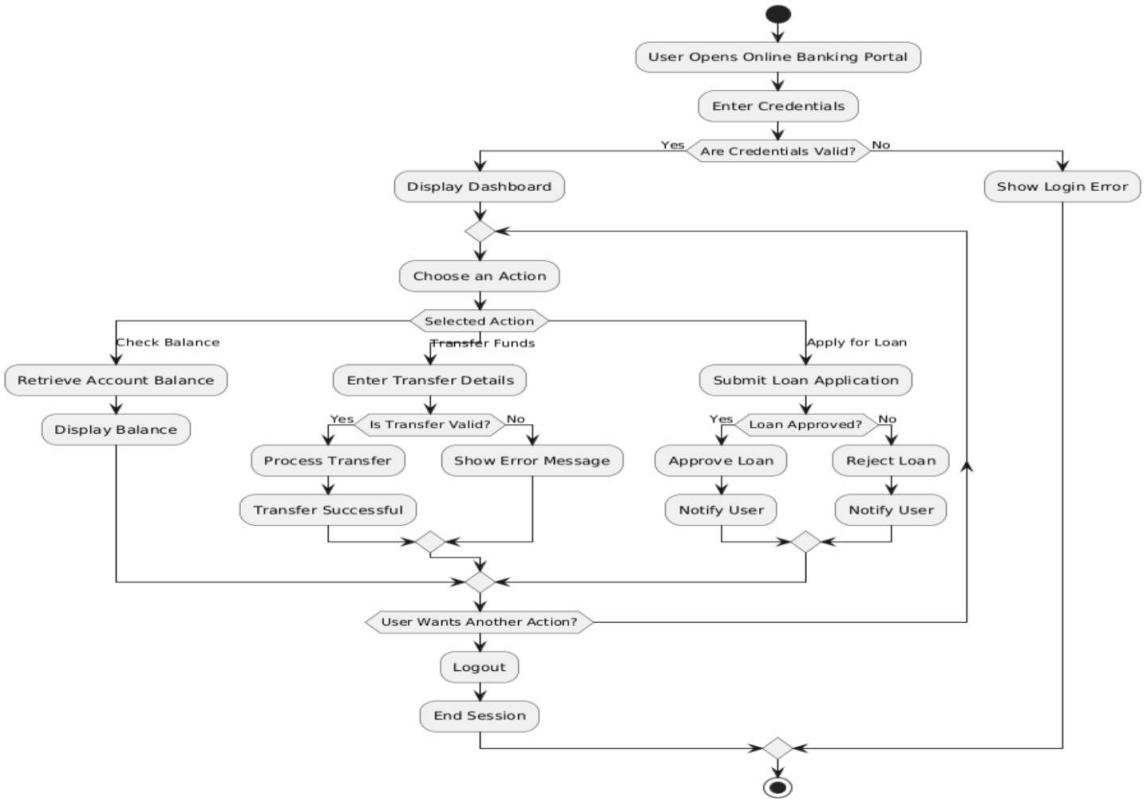
else (No)

:Show Login Error;

endif

stop

@enduml



COMPONENT DIAGRAM

@startuml

component "User Interface" as UI

component "Authentication Service" as Auth

component "Banking Services" as Banking

component "Transaction Service" as Transaction

component "Loan Processing" as Loan

component "Database" as DB

UI --> Auth : User Login

Auth --> DB : Verify Credentials

Auth --> UI : Authentication Success/Failure

UI --> Banking : Request Account Info

Banking --> DB : Fetch Account Details

Banking --> UI : Display Account Details

UI --> Transaction : Transfer Funds

Transaction --> Banking : Validate Funds

Transaction --> DB : Update Account Balance

Transaction --> UI : Show Transaction Status

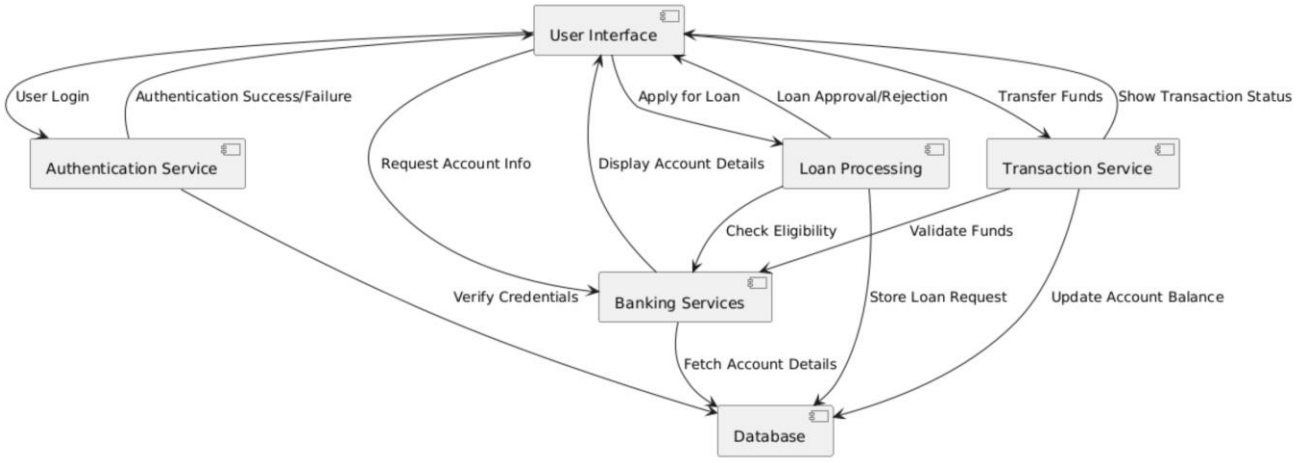
UI --> Loan : Apply for Loan

Loan --> Banking : Check Eligibility

Loan --> DB : Store Loan Request

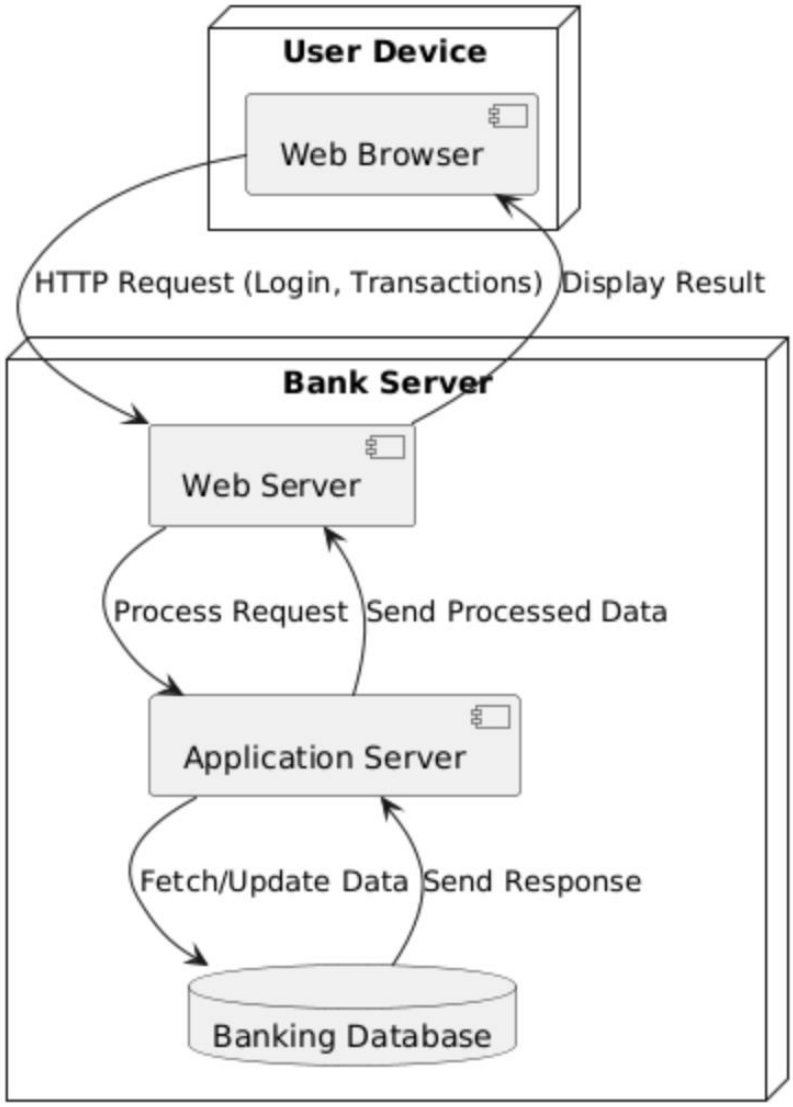
Loan --> UI : Loan Approval/Rejection

@enduml



DEPLOYMENT DIAGRAM

```
@startuml
node "User Device" {
   component "Web Browser" as Browser
}
node "Bank Server" {
   component "Web Server" as WebServer
   component "Application Server" as AppServer
   database "Banking Database" as DB
}
Browser --> WebServer : HTTP Request (Login, Transactions)
WebServer --> AppServer : Process Request
AppServer --> DB : Fetch/Update Data
DB --> AppServer : Send Response
AppServer --> WebServer : Send Processed Data
WebServer --> Browser : Display Result
@enduml
```


User Device
Web Browser
HTTP Request (Login, Transactions)
Display Result
Bank Server
Web Server
Process Request
Send Processed Data
Application Server
Fetch/Update Data
Send Response
Banking Database