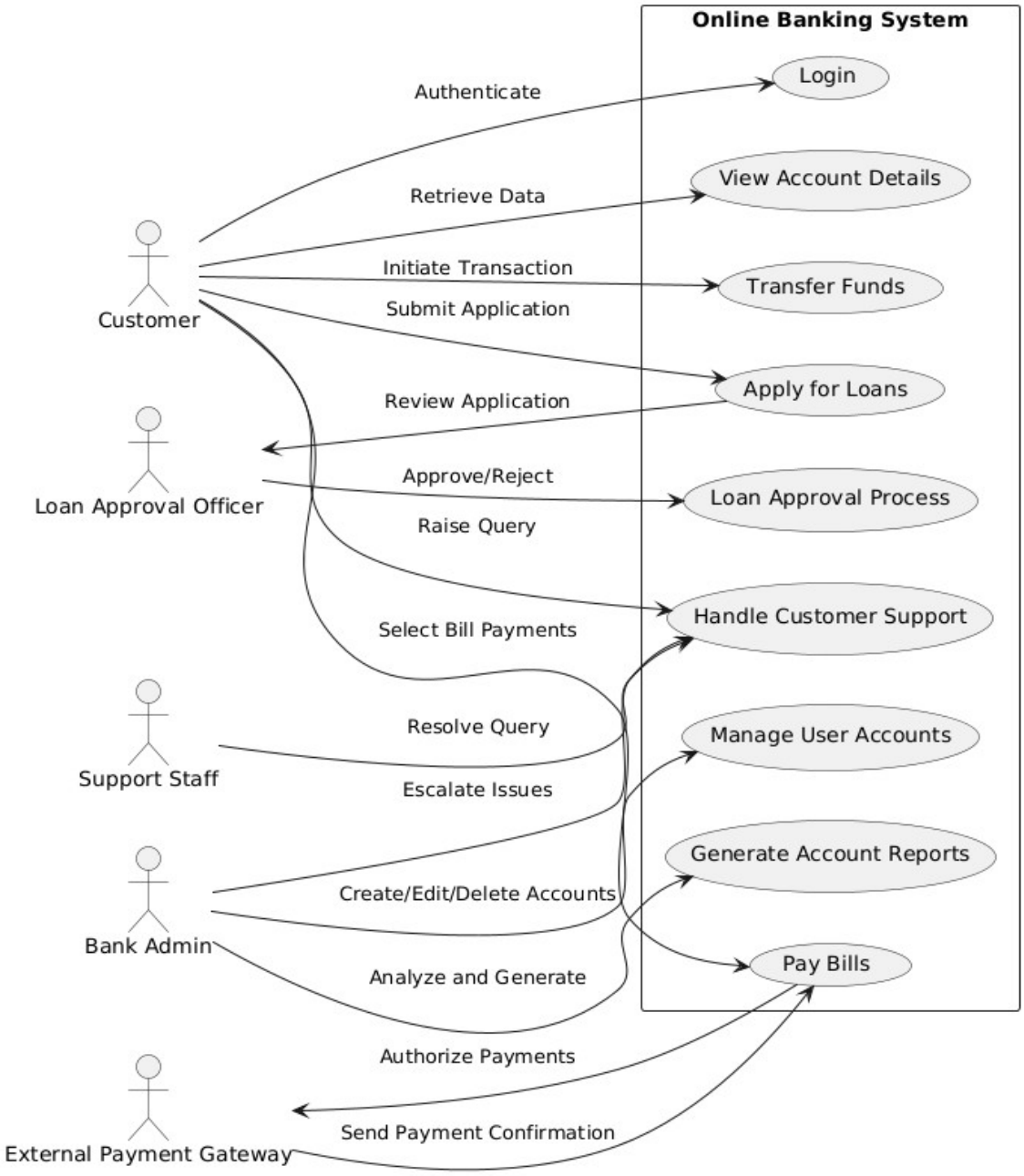
Experiment-04
UML diagrams for Online Banking System

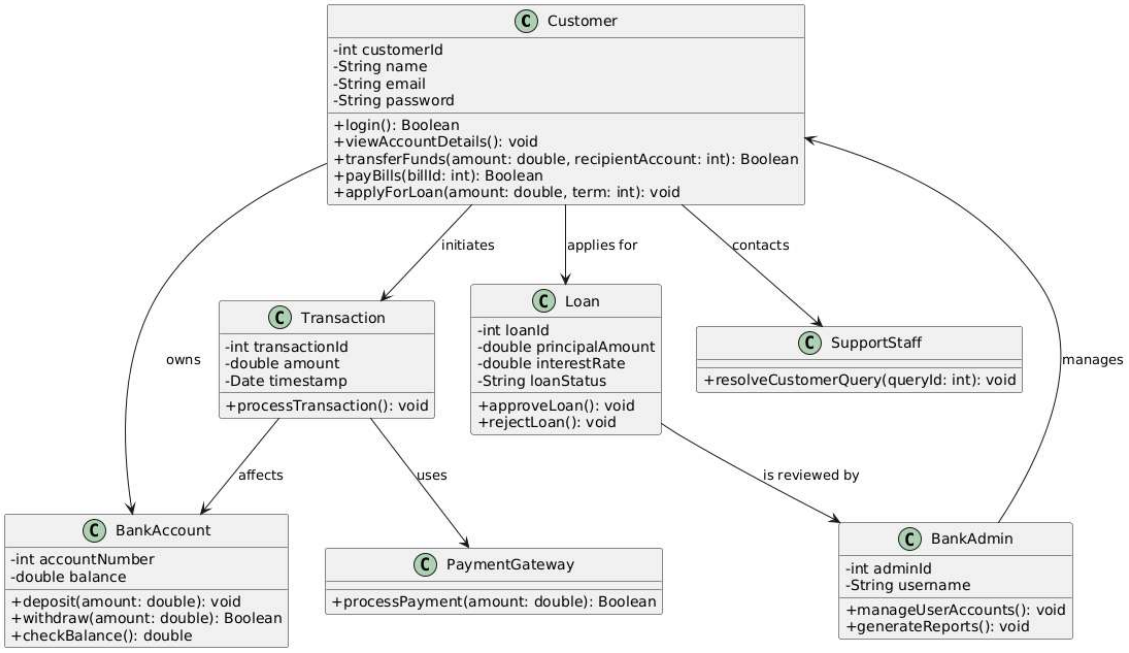
1.Use-Case Diagram



```
@startuml
left to right direction
skinparam packageStyle rectangle
actor Customer as C
actor "Bank Admin" as BA
actor "External Payment Gateway" as EPG
actor "Loan Approval Officer" as LAO
actor "Support Staff" as SS

rectangle "Online Banking System" {
    (Login) as UC1
    (View Account Details) as UC2
    (Transfer Funds) as UC3
    (Pay Bills) as UC4
    (Apply for Loans) as UC5
    (Loan Approval Process) as UC6
    (Manage User Accounts) as UC7
    (Generate Account Reports) as UC8
    (Handle Customer Support) as UC9
}

' Customer Use Cases
C --> UC1 : "Authenticate"
C --> UC2 : "Retrieve Data"
C --> UC3 : "Initiate Transaction"
C --> UC4 : "Select Bill Payments"
C --> UC5 : "Submit Application"
C --> UC9 : "Raise Query"
```

```
' Admin Use Cases
BA --> UC7 : "Create/Edit/Delete Accounts"
BA --> UC8 : "Analyze and Generate"
BA --> UC9 : "Escalate Issues"

' Payment Gateway Integration
UC4 --> EPG : "Authorize Payments"
EPG --> UC4 : "Send Payment Confirmation"

' Loan Approval Process
UC5 --> LAO : "Review Application"
LAO --> UC6 : "Approve/Reject"

' Support Staff Use Case
SS --> UC9 : "Resolve Query"
@enduml
```

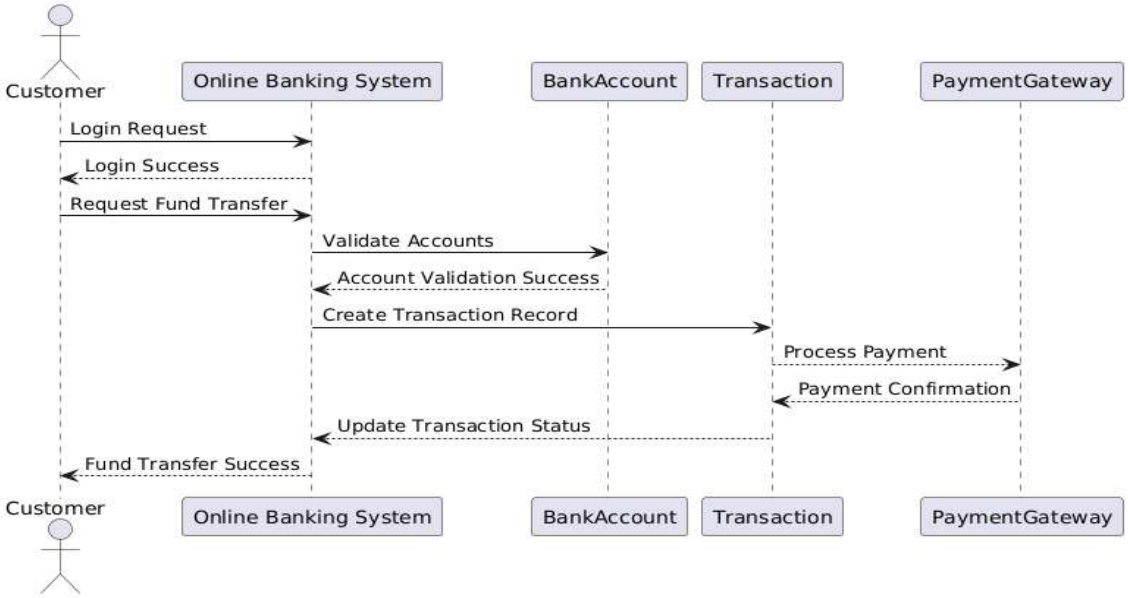
2.Class Diagram



```
@startuml
' Advanced Class Diagram for Online Banking System
skinparam classAttributeIconSize 0

class Customer {
    - int customerId
    - String name
    - String email
    - String password
    + login(): Boolean
    + viewAccountDetails(): void
    + transferFunds(amount: double, recipientAccount: int): Boolean
    + payBills(billId: int): Boolean
    + applyForLoan(amount: double, term: int): void
}

class BankAccount {
    - int accountNumber
    - double balance
    + deposit(amount: double): void
    + withdraw(amount: double): Boolean
    + checkBalance(): double
}

class Transaction {
    - int transactionId
    - double amount
    - Date timestamp
    + processTransaction(): void
}
```

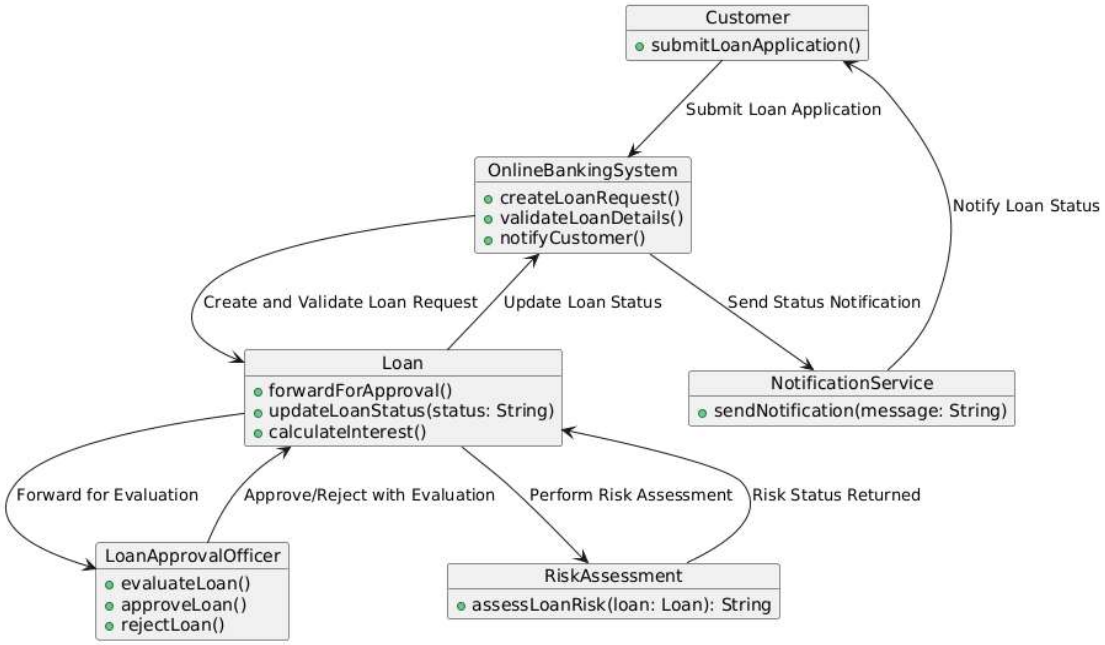
```
class Loan {
    - int loanId
    - double principalAmount
    - double interestRate
    - String loanStatus
    + approveLoan(): void
    + rejectLoan(): void
}

class BankAdmin {
    - int adminId
    - String username
    + manageUserAccounts(): void
    + generateReports(): void
}

class PaymentGateway {
    + processPayment(amount: double): Boolean
}

class SupportStaff {
    + resolveCustomerQuery(queryId: int): void
}
Customer --> BankAccount : "owns"
Customer --> Transaction : "initiates"
Customer --> Loan : "applies for"
Customer --> SupportStaff : "contacts"
Transaction --> BankAccount : "affects"
BankAdmin --> Customer : "manages"
Loan --> BankAdmin : "is reviewed by"
Transaction --> PaymentGateway : "uses"
@enduml
```

3.Sequence Diagram




```
@startuml
' Advanced Sequence Diagram for Fund Transfer
actor Customer
participant "Online Banking System" as OBS
participant BankAccount
participant Transaction
participant PaymentGateway

Customer -> OBS: Login Request
OBS --> Customer: Login Success
Customer -> OBS: Request Fund Transfer
OBS -> BankAccount: Validate Accounts
BankAccount --> OBS: Account Validation Success
OBS -> Transaction: Create Transaction Record
Transaction --> PaymentGateway: Process Payment
PaymentGateway --> Transaction: Payment Confirmation
Transaction --> OBS: Update Transaction Status
OBS --> Customer: Fund Transfer Success
@enduml
```

4. Collaboration Diagram




```
@startuml

' Enhanced Collaboration Diagram for Loan Application with Extended
Functionality

object Customer {

    + submitLoanApplication()

}

object OnlineBankingSystem {

    + createLoanRequest()

    + validateLoanDetails()

    + notifyCustomer()

}

object Loan {

    + forwardForApproval()

    + updateLoanStatus(status: String)

    + calculateInterest()

}
```

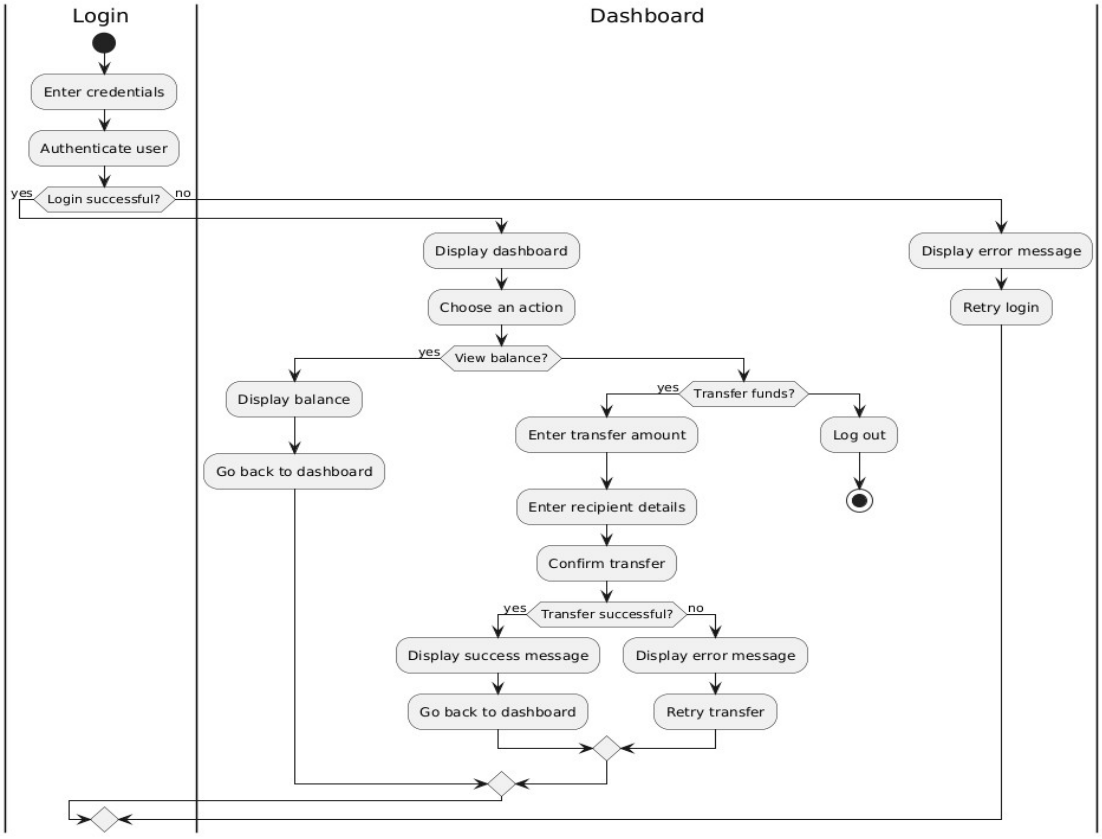
```
object LoanApprovalOfficer {
    + evaluateLoan()
    + approveLoan()
    + rejectLoan()
}
object RiskAssessment {
    + assessLoanRisk(loan: Loan): String
}
object NotificationService {
    + sendNotification(message: String)
}

Customer --> OnlineBankingSystem : "Submit Loan Application"
OnlineBankingSystem --> Loan : "Create and Validate Loan Request"
Loan --> RiskAssessment : "Perform Risk Assessment"
RiskAssessment --> Loan : "Risk Status Returned"
Loan --> LoanApprovalOfficer : "Forward for Evaluation"
LoanApprovalOfficer --> Loan : "Approve/Reject with Evaluation"
Loan --> OnlineBankingSystem : "Update Loan Status"
OnlineBankingSystem --> NotificationService : "Send Status
Notification"
NotificationService --> Customer : "Notify Loan Status"
@enduml
```

5. State-Chart Diagram


```
@startuml
[*] --> InitialState

InitialState --> Login : User enters credentials
Login --> Dashboard : Successful login
Login --> AccountLocked : Too many failed login attempts

AccountLocked --> AccountLocked : Wait for lock period to expire
AccountLocked --> Login : Retry after lock period

Dashboard --> ViewBalance : View account balance
Dashboard --> TransferFunds : Transfer funds
Dashboard --> Logout : Logout
```

```
ViewBalance --> Dashboard : Return to dashboard
TransferFunds --> EnterAmount : Enter transfer amount
EnterAmount --> ConfirmTransfer : Confirm transfer details
ConfirmTransfer --> TransferSuccess : Transfer successful
TransferSuccess --> Dashboard : Return to dashboard

Logout --> InitialState : Return to initial state
Logout --> [*] : End session
@enduml
```

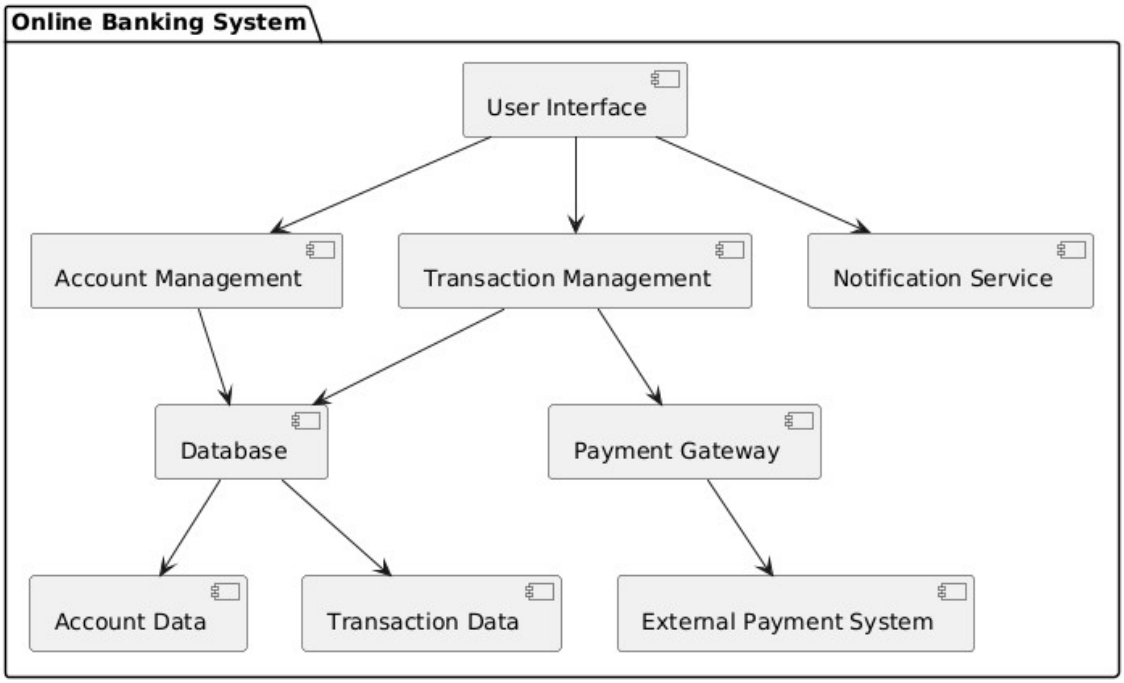
6. Activity Diagram




```
@startuml
|Login|
start
:Enter credentials;
```

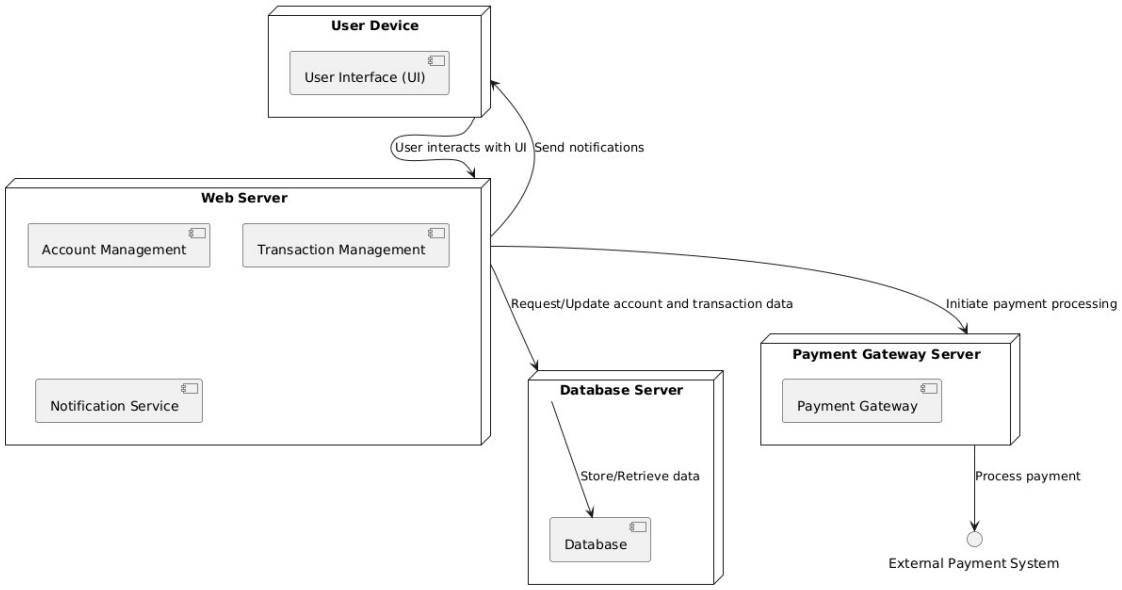
```
:Authenticate user;
if (Login successful?) then (yes)
  |Dashboard|
  :Display dashboard;
  :Choose an action;
  if (View balance?) then (yes)
    :Display balance;
    :Go back to dashboard;
  else
    if (Transfer funds?) then (yes)
      :Enter transfer amount;
      :Enter recipient details;
      :Confirm transfer;
      if (Transfer successful?) then (yes)
        :Display success message;
        :Go back to dashboard;
      else (no)
        :Display error message;
        :Retry transfer;
      endif
    else
      :Log out;
      stop
    endif
  endif
else (no)
  :Display error message;
  :Retry login;
endif
@enduml
```

7. Component Diagram




```
@startuml
package "Online Banking System" {

  [User Interface] -down-> [Account Management]
  [User Interface] -down-> [Transaction Management]
  [User Interface] -down-> [Notification Service]

  [Account Management] -down-> [Database]
  [Transaction Management] -down-> [Payment Gateway]
  [Transaction Management] -down-> [Database]

  [Database] --> [Account Data]
  [Database] --> [Transaction Data]

  [Payment Gateway] --> [External Payment System]
}
@enduml
```

8. Deployment Diagram




```
@startuml
node "User Device" {
  component "User Interface (UI)"
}

node "Web Server" {
  component "Account Management"
  component "Transaction Management"
  component "Notification Service"
}

node "Database Server" {
  component "Database"
}

node "Payment Gateway Server" {
  component "Payment Gateway"
}
```

```
"User Device" --> "Web Server" : User interacts with UI

"Web Server" --> "Database Server" : Request/Update account and
transaction data

"Web Server" --> "Payment Gateway Server" : Initiate payment
processing

"Web Server" --> "User Device" : Send notifications

"Database Server" --> "Database" : Store/Retrieve data

"Payment Gateway Server" --> "External Payment System" : Process
payment

@enduml
```