BELGIUM CAMPUS ITVERSITY

SOFTWARE-TECH SOLUTIONS

REPORT

Prepared by :
Aphelele Mdebuka (601706)

Prepared by :
Phaka Molokomme (601993)

Prepared by :
Phenyo Phooko (602218)

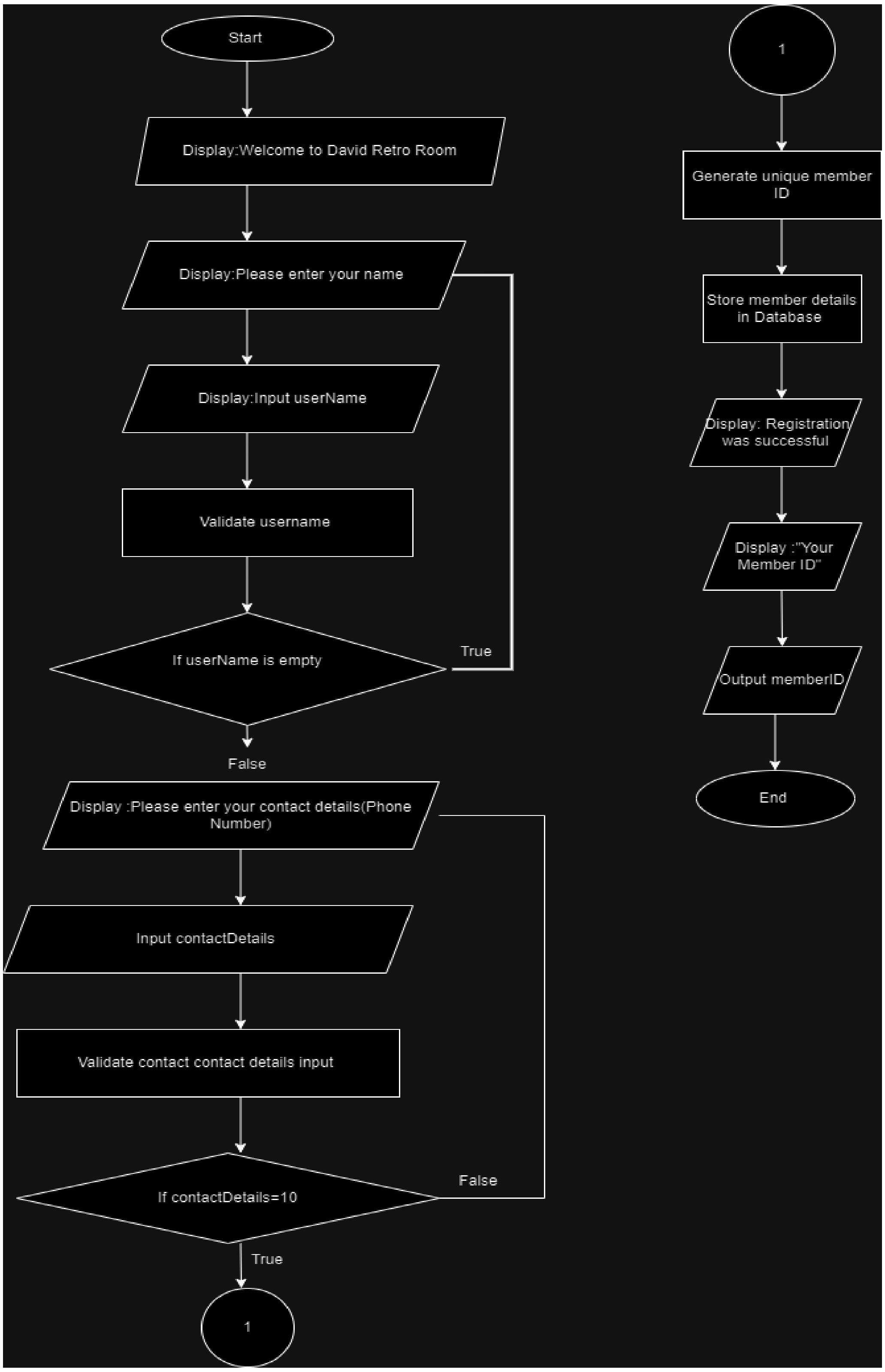
REGISTRATION

In this pseudocode:

- The `registerMember` function prompts the user to enter their name and contact details.
- It validates the name input to ensure it is not empty.
- It validates the contact details input to ensure it is an integer and has a length of 10 digits.
- It generates a unique member ID.
- It stores the member details (name and contact details) along with the generated member ID in the database.
- Finally, it displays a confirmation message along with the generated member ID.

```
BEGIN
SET registerMember
  DISPLAY "Welcome to David's Retro Room Registration Portal"
  DISPLAY "Please enter your name:"
  userName = INPUT

  WHILE userName is empty DO
    DISPLAY "Name cannot be empty. Please enter your name:"

  DISPLAY "Please enter your contact details (phone number):"
  contactDetails = INPUT

  WHILE contactDetails is not an integer or length of contactDetails is not = 10 DO
    DISPLAY "Invalid phone number format. Please enter a 10-digit phone number:"

  memberId = generateUniqueID

  memberData =
    "ID": memberId,
        "Name": userName,
        "ContactDetails": contactDetails

    Save memberData to database

    DISPLAY "Registration successful!"
    DISPLAY "Your Member ID is: " + memberId

    RETURN memberId
END
```

Flowchart 1


Start
Display:Welcome to David Retro Room
Display:Please enter your name
Display:Input userName
Validate username
If userName is empty
True
False
Display :Please enter your contact details(Phone Number)
Input contactDetails
Validate contact contact details input
If contactDetails=10
False
True
1
1
Generate unique member ID
Store member details in Database
Display: Registration was successful
Display :"Your Member ID"
Output memberID
End

MEMBER MANAGEMENT

In this pseudocode:

- The `updateMemberDetails` function takes the member ID, updated name, and updated contact information as input parameters.

- It retrieves the existing member details from the database using the provided member ID.

- It updates the member details with the provided updated name and/or contact information, checking if each field is provided in the `updatedName` and `updatedContactInfo` parameters.

- If updated contact information is provided, it validates the input to ensure it is an integer and has a length of 10 digits.

- After updating the member details, it saves the updated information back to the database.

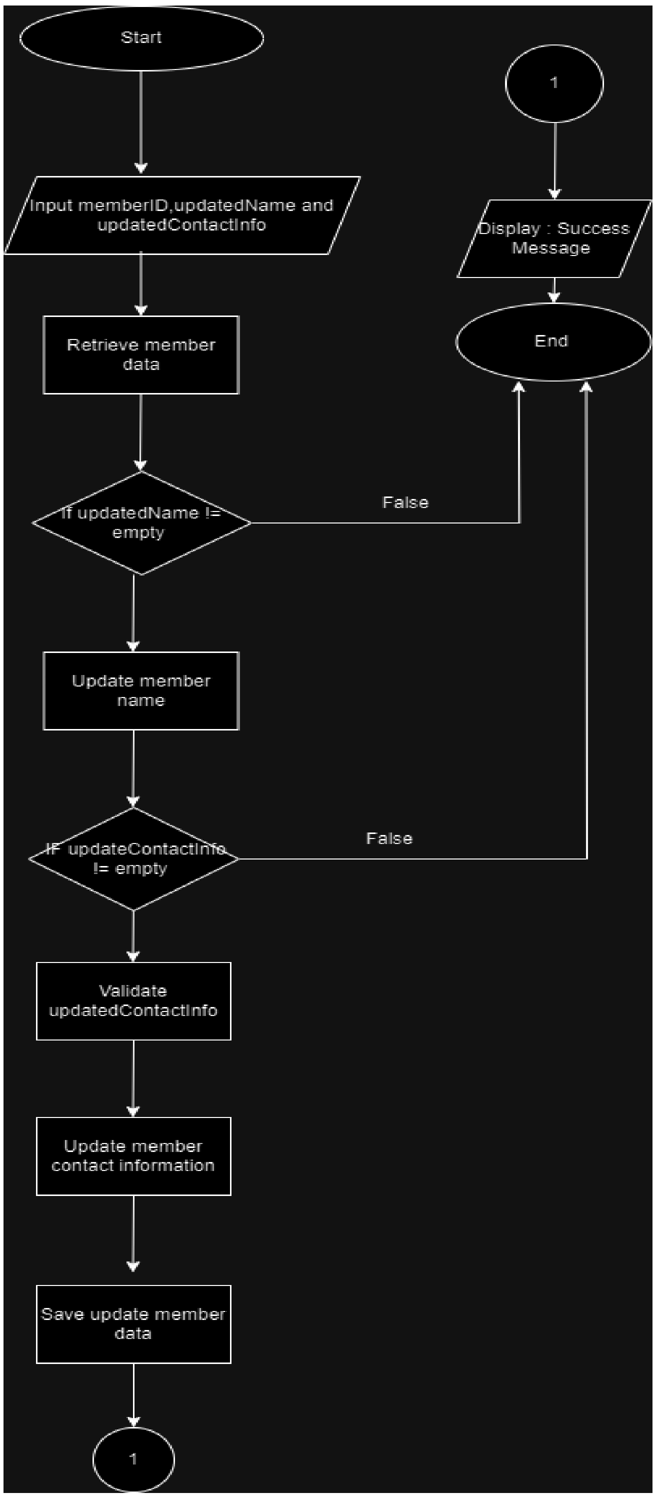
- Finally, it displays a confirmation message to indicate that the member details have been updated successfully.

The functions `retrieveMemberData` and `saveUpdatedMemberData` are placeholders for the actual database retrieval and update operations, respectively. These functions need to be implemented based on the database system and programming language being used.

```
BEGIN
SET updateMemberDetails(memberId, updatedName, updatedContactInfo)
  memberId = INPUT
  updatedName = INPUT
   updatedContactInfo = INPUT

   memberData = retrieveMemberData(memberId)

   IF updatedName is not empty:
      memberData.name = updatedName
   IF updatedContactInfo is not empty:


      WHILE updatedContactInfo is not an integer or length of updatedContactInfo is not = 10 DO
         DISPLAY "Invalid phone number format. Please enter a 10-digit phone number:"
      memberData.contactInfo = updatedContactInfo
```

```
    saveUpdatedMemberData(memberId, memberData)

    DISPLAY "Member details updated successfully."

END
```

Flowchart 2


Start
Input memberID,updatedName and updatedContactInfo
Retrieve member data
If updatedName != empty
False
Update member name
IF updateContactInfo != empty
False
Validate updatedContactInfo
Update member contact information
Save update member data
1
1
Display : Success Message
End

MEMBER PORTAL

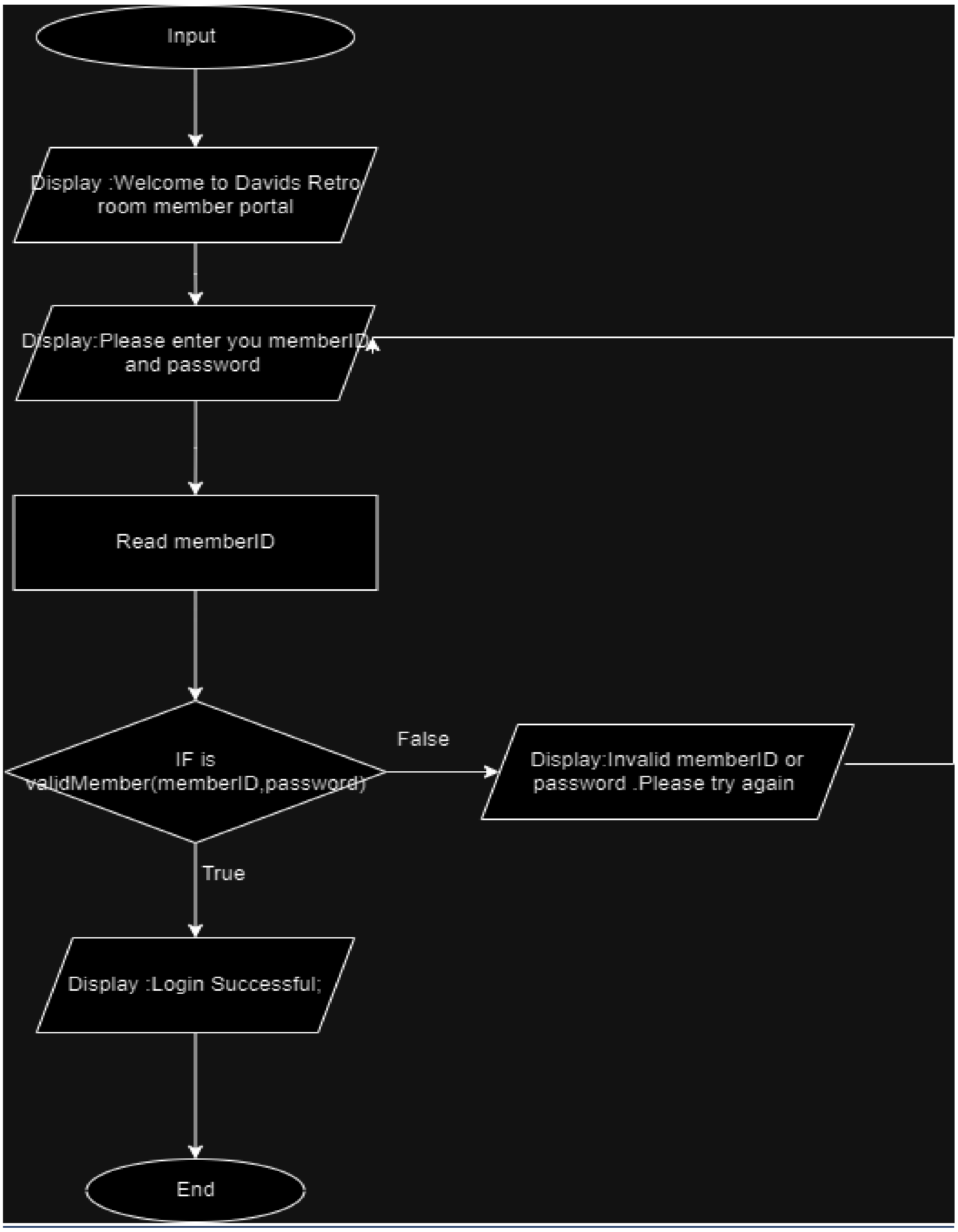
In this pseudocode:

- The `memberPortalLogin` function prompts the user to enter their member ID and password.
- It then calls the `isValidMember` function to validate the provided member ID and password.
- If the member ID and password are valid, it displays a success message and returns True.
- If the member ID and password are invalid, it displays an error message and returns False.

You would need to implement the `isValidMember` function separately to perform the actual validation against your database or authentication system. This function would typically query the database to check if the provided member ID exists and if the corresponding password matches the stored password for that member ID.

```
BEGIN

SET memberPortalLogin
   DISPLAY "Welcome to David's Retro Room Member Portal"
   DISPLAY "Please enter your Member ID:"
      memberId = INPUT
   DISPLAY "Please enter your password:"
      password = INPUT

   IF isValidMember(memberId, password):
      DISPLAY "Login successful!"
      RETURN True
   ELSE:
      DISPLAY "Invalid Member ID or password. Please try again."
      RETURN False

END
```

Flowchart 3


Input
Display :Welcome to Davids Retro room member portal
Display:Please enter you memberID and password
Read memberID
IF is validMember(memberID,password)
False
Display:Invalid memberID or password .Please try again
True
Display :Login Successful;
End

COLLECTIBLE MANAGEMENT

In this pseudocode:

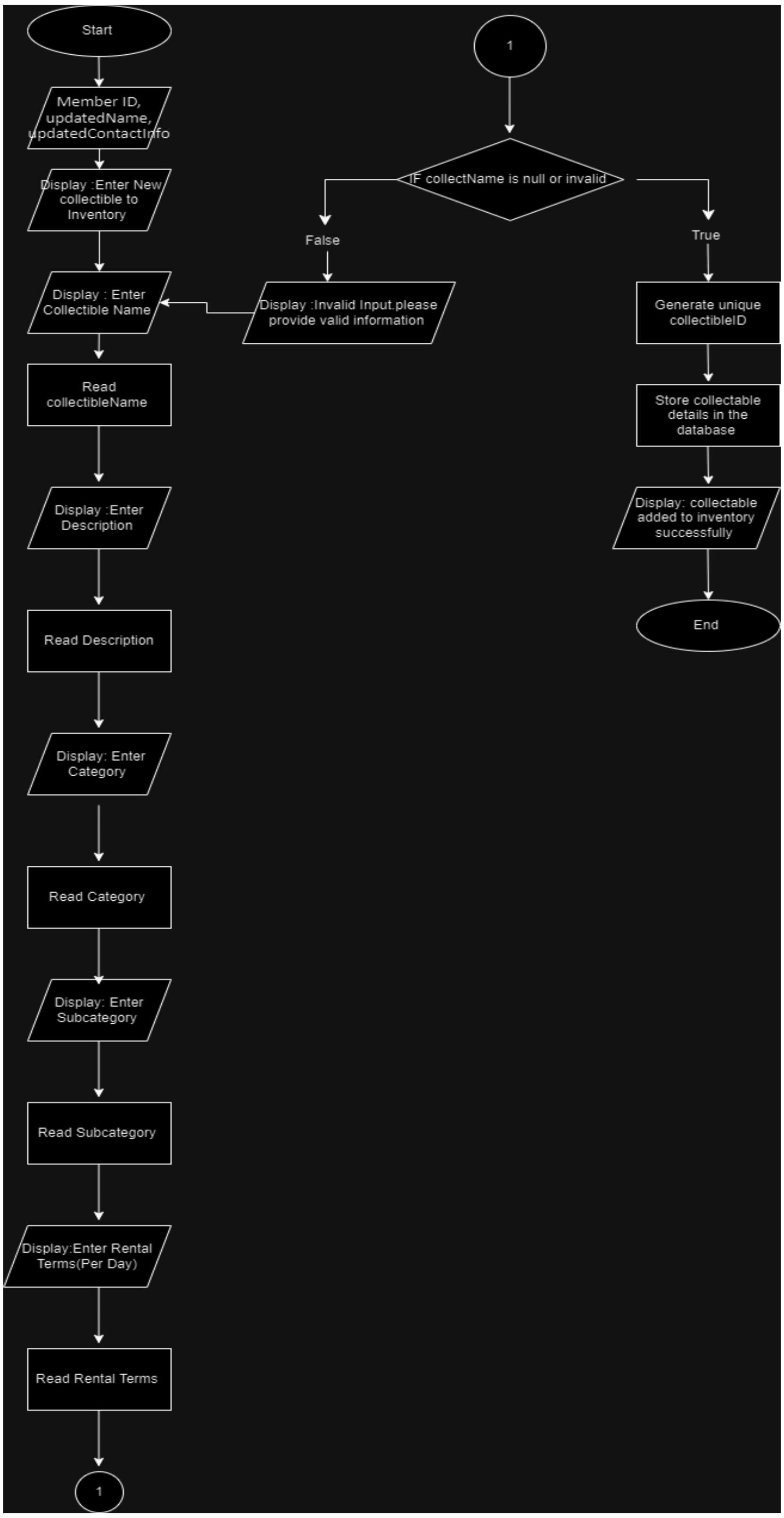
- The `addCollectible` function allows the user to add a new collectible to the inventory by providing details such as name, description, category, subcategory, and rental terms. It validates the input and stores the collectible details in the database, setting the availability status to true by default.

- The `checkCollectibleAvailability` function takes a collectible ID as input and checks its availability by retrieving its details from the database. If the collectible exists and its availability status is true, it returns true, indicating that the collectible is available; otherwise, it returns false.

You would need to implement the `generateUniqueID`, `retrieveCollectibleData`, and `saveCollectibleData` functions for generating unique IDs, retrieving collectible data from the database, and saving collectible data to the database, respectively. Additionally, you may need to adjust the database schema and queries based on your specific requirements and database system.

```
BEGIN

SET addCollectible
  DISPLAY "Add New Collectible to Inventory:"
  DISPLAY "Enter Collectible Name:"
  collectibleName = INPUT
  DISPLAY "Enter Description:"
  description = INPUT
  DISPLAY "Enter Category:"
  category = INPUT
  DISPLAY "Enter Subcategory:"
  subcategory
  DISPLAY "Enter Rental Terms (per day):"
  rentalTerms = INPUT

 IF collectibleName is empty or description is empty or category is empty or subcategory is empty or
rentalTerms is not an integer:
    DISPLAY "Invalid input. Please provide valid information."
    RETURN
    collectibleId = generateUniqueID

    collectibleData =
        "ID": collectibleId,
        "Name": collectibleName,
        "Description": description,
        "Category": category,
        "Subcategory": subcategory,
        "RentalTerms": rentalTerms,
        "Availability": true // Set availability to true by default

    Save collectibleData to database

    DISPLAY "Collectible added to inventory successfully."

END


SET checkCollectibleAvailability(collectibleId):

    collectibleData = retrieveCollectibleData(collectibleId)

    IF collectibleData is not null and collectibleData.Availability is true:
        DISPLAY "Collectible is available."
        RETURN True
    ELSE:
        DISPLAY "Collectible is not available."
        RETURN False

END
```

Flowchart 4



Start
Member ID, updatedName, updatedContactInfo
Display :Enter New collectible to Inventory
Display : Enter Collectible Name
Read collectibleName
Display :Enter Description
Read Description
Display: Enter Category
Read Category
Display: Enter Subcategory
Read Subcategory
Display:Enter Rental Terms(Per Day)
Read Rental Terms
1
1
IF collectName is null or invalid
False
True
Display :Invalid Input.please provide valid information
Generate unique collectibleID
Store collectable details in the database
Display: collectable added to inventory successfully
End

SALES MANAGEMENT

In this pseudocode:

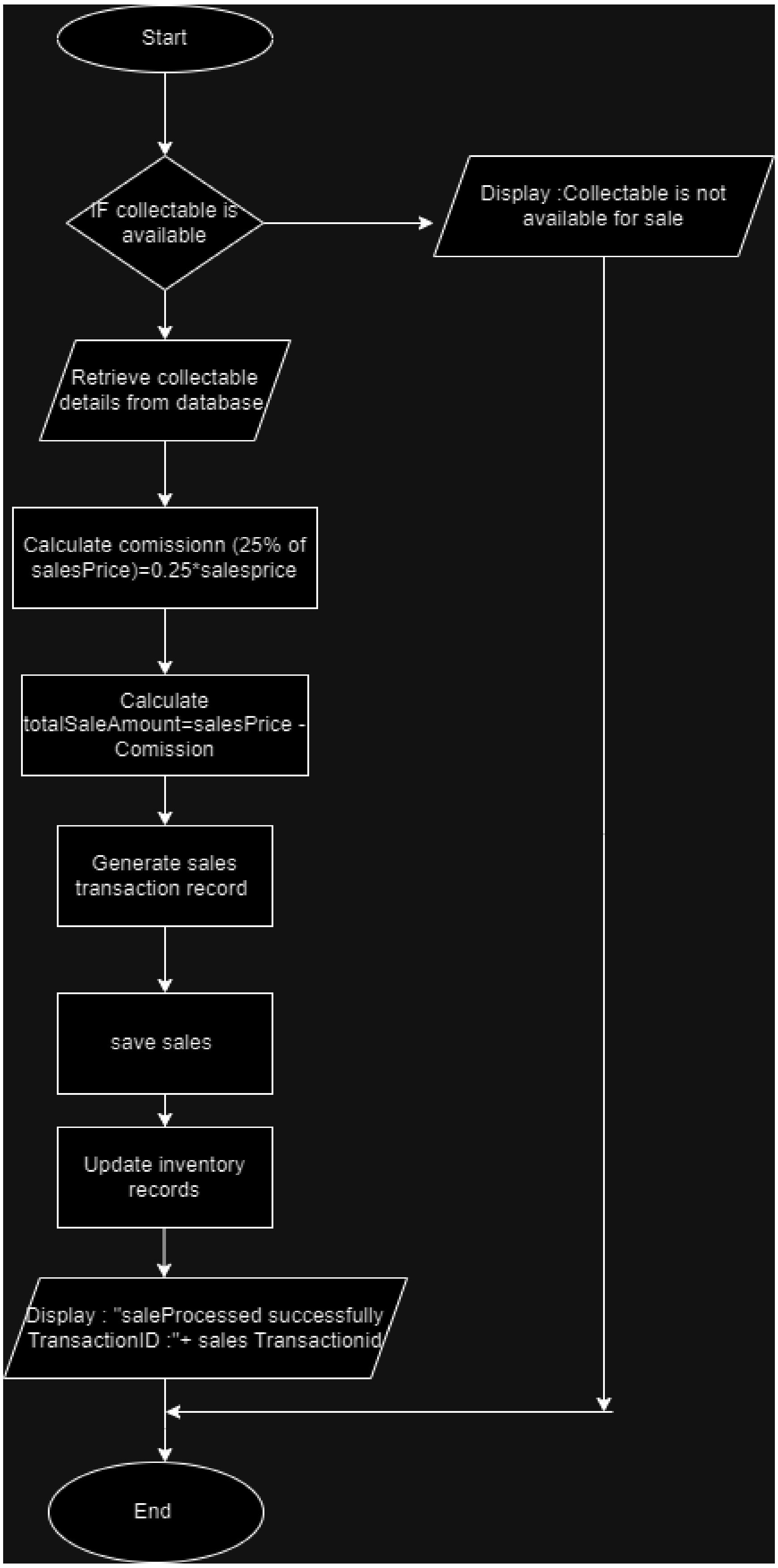
- The `processSale` function takes the collectible ID and sales price as input parameters.
- It first checks if the collectible is available for sale using the `isCollectibleAvailable` function.
- If the collectible is available, it retrieves its details from the database and updates its availability status to false.
- It computes the commission as 25% of the sales price.
- It calculates the total sale amount by deducting the commission from the sales price.
- It generates a unique sales transaction ID, creates a sales transaction record, and saves it to the database.
- It updates inventory records to reflect the sale.
- Finally, it displays a confirmation message with the generated sales transaction ID if the sale is processed successfully.

Adjustments have been made to include commission calculations and the total sale amount. You would still need to implement the functions `isCollectibleAvailable`, `retrieveCollectibleData`, `updateCollectibleAvailability`, `generateUniqueTransactionID`, `saveSalesTransaction`, and `updateInventoryRecords` according to your specific requirements and database system.

```
BEGIN
SET processSale(collectibleId, salesPrice):
salesPrice = INPUT

  IF isCollectibleAvailable(collectibleId):
    collectibleData = retrieveCollectibleData(collectibleId)

    updateCollectibleAvailability(collectibleId, false)

    commission = salesPrice * 0.25
```

```
        totalSaleAmount = salesPrice - commission

        salesTransactionId = generateUniqueTransactionID
        salesTransaction =
            "TransactionID": salesTransactionId,
            "CollectibleID": collectibleId,
            "SalesPrice": salesPrice,
            "Commission": commission,
            "TotalSaleAmount": totalSaleAmount,
            "Date": currentDate

        saveSalesTransaction(salesTransaction)

        updateInventoryRecords(collectibleId)

        DISPLAY "Sale processed successfully. Transaction ID: " + salesTransactionId
    ELSE:
        DISPLAY "Collectible is not available for sale."

END
```

Flowchart 5


Start
IF collectable is available
Display :Collectable is not available for sale
Retrieve collectable details from database
Calculate comissionn (25% of salesPrice)=0.25*salesprice
Calculate totalSaleAmount=salesPrice - Comission
Generate sales transaction record
save sales
Update inventory records
Display : "saleProcessed successfully TransactionID :"+ sales Transactionid
End

REPORTING & ANALYTICS

In this pseudocode:

- The `generateAnalyticsReport` function takes a date range and analytical parameters as input.

- It first retrieves sales data from the database within the specified date range using the `retrieveSalesData` function.

- Depending on the provided analytical parameters, it performs different types of analytics.

- If the analytical parameters indicate customer trends, it calls the `analyzeCustomerTrends` function to generate a customer trends report.

- If the analytical parameters indicate inventory turnover, it calls the `calculateInventoryTurnover` function to calculate inventory turnover.

- If the analytical parameters indicate sales performance, it calls the `analyzeSalesPerformance` function to analyze sales performance.

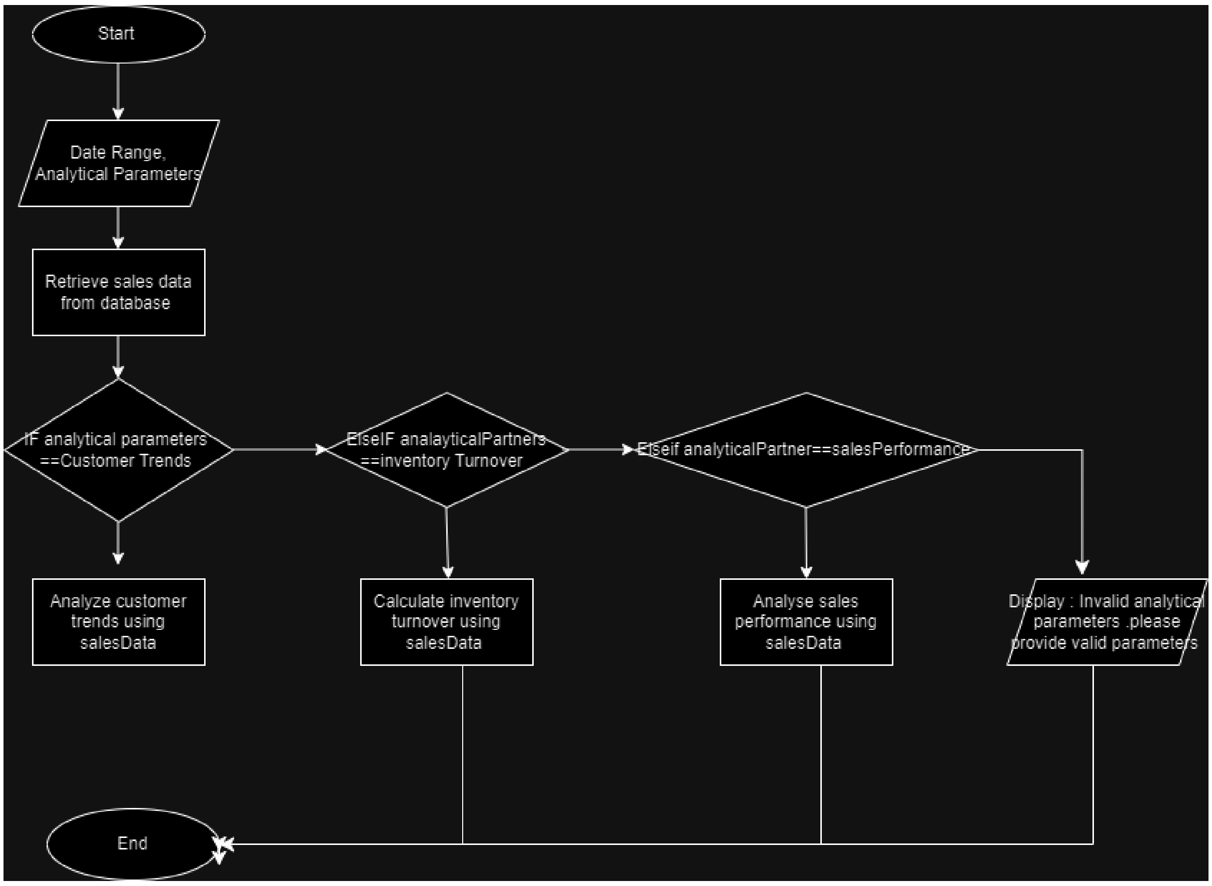
- Finally, it returns the generated analytics report based on the specified parameters.

You would need to implement the functions `retrieveSalesData`, `analyzeCustomerTrends`, `calculateInventoryTurnover`, and `analyzeSalesPerformance` to retrieve sales data and perform the corresponding analytics according to your specific requirements and database system. Additionally, you may need to adjust the analytical parameters and report generation logic based on your business needs.

```
BEGIN
SET generateAnalyticsReport(dateRange, analyticalParameters):

  dateRange = INPUT
  salesData = retrieveSalesData(dateRange)

  IF analyticalParameters == "customerTrends":
    customerTrendsReport = analyzeCustomerTrends(salesData)
    RETURN customerTrendsReport
  ELSE IF analyticalParameters == "inventoryTurnover":
    inventoryTurnoverReport = calculateInventoryTurnover(salesData)
    RETURN inventoryTurnoverReport
  ELSE IF analyticalParameters == "salesPerformance":
    salesPerformanceReport = analyzeSalesPerformance(salesData)
```

```
        RETURN salesPerformanceReport
    ELSE:
        DISPLAY "Invalid analytical parameters. Please provide valid parameters."

END
```

Flowchart 6


Start
Date Range, Analytical Parameters
Retrieve sales data from database
IF analytical parameters ==Customer Trends
ElseIF analaylticalPartners ==inventory Turnover
Elseif analyticalPartner==salesPerformance
Analyze customer trends using salesData
Calculate inventory turnover using salesData
Analyse sales performance using salesData
Display : Invalid analytical parameters .please provide valid parameters
End

AUTOMATED NOTIFCATIONS

In this pseudocode:

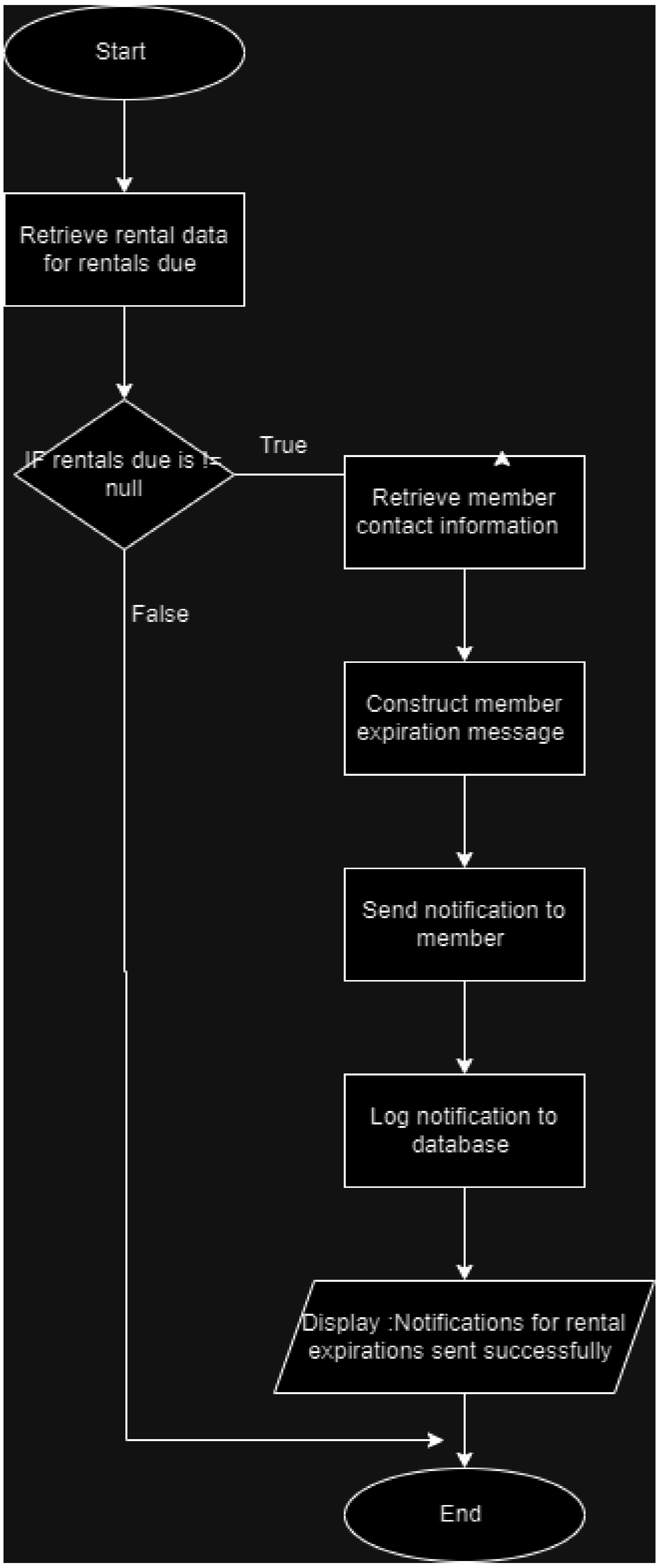
- The `sendNotificationsForImminentRentalExpirations` function retrieves rental data for rentals due to expire in 24 hours from the database.
- It iterates through the rentals due to expire and sends notifications to the respective members.
- For each rental due to expire, it constructs a notification message reminding the member about the upcoming expiration date and sends it.
- The notifications are sent to the respective members' email addresses retrieved from the database.
- After sending notifications, a confirmation message is displayed.

You would need to implement the functions `retrieveRentalsDueToExpire`, `retrieveMemberContactInfo`, `sendNotification`, `logNotification`, and `currentDate` according to your specific requirements and available technologies. Additionally, you may need to adjust the notification content and delivery method based on your business needs.

```
BEGIN
SET sendNotificationsForImminentRentalExpirations

  rentalsDueToExpire = retrieveRentalsDueToExpire

  FOR each rental in rentalsDueToExpire:
    memberId = rental.memberId
    collectibleId = rental.collectibleId
    expirationDate = rental.expirationDate

    memberContactInfo = retrieveMemberContactInfo(memberId)

    notificationMessage = "Dear [Member Name], your rental for collectible [Collectible Name] is due to expire in 24 hours on [Expiration Date]. Please return it or renew the rental if needed. Thank you."

    sendNotification(memberContactInfo.email, notificationMessage)
```

```
        logNotification(memberId, collectibleId, notificationMessage, currentDate)

    DISPLAY "Notifications for imminent rental expirations sent successfully."

END
```

Flowchart 7


Start
Retrieve rental data for rentals due
IF rentals due is != null
True
False
Retrieve member contact information
Construct member expiration message
Send notification to member
Log notification to database
Display :Notifications for rental expirations sent successfully
End

RENTAL CALCULATIONS

In this pseudocode:

- The `calculateRent` function takes the collectible ID and rental duration as inputs.

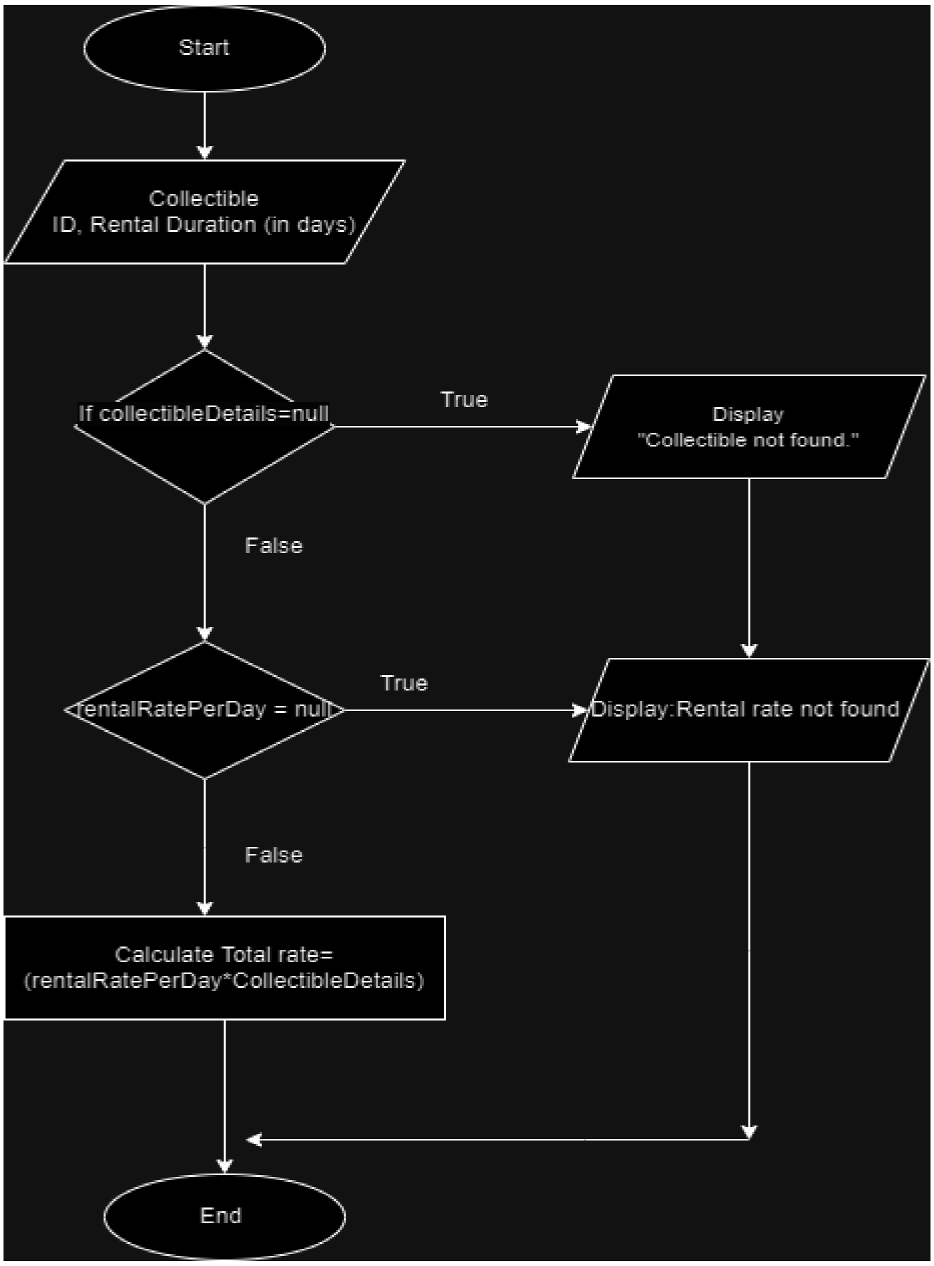
- It retrieves collectible details such as rental rate per day from the database using the collectible ID.

- Then, it calculates the total rent amount by multiplying the rental rate per day by the rental duration.

You would need to implement the functions `retrieveCollectibleDetails` and `getRentalRatePerDay` to retrieve collectible details and rental rates from the database based on the collectible ID. Additionally, adjust the rental calculation logic based on your business requirements and pricing structure.

```
BEGIN
SET calculateRent(collectibleId, rentalDuration):
  rentalDuration = INPUT
  collectibleDetails = retrieveCollectibleDetails(collectibleId)

  IF collectibleDetails is null:
    DISPLAY "Collectible not found."
    RETURN 0

  rentalRatePerDay = getRentalRatePerDay(collectibleId)

  IF rentalRatePerDay is null:
    DISPLAY "Rental rate not found."
    RETURN 0

  totalRentAmount = rentalRatePerDay * rentalDuration(collectableDetails)

  RETURN totalRentAmount

END
```

Flowchart 8


Start
Collectible
ID, Rental Duration (in days)
If collectibleDetails=null
True
Display
"Collectible not found."
False
rentalRatePerDay = null
True
Display:Rental rate not found
False
Calculate Total rate=
(rentalRatePerDay*CollectibleDetails)
End