# 1. SYSTEM INITIALIZATION

```
BEGIN SYSTEM

    LOAD database connection
    INITIALIZE tables: Users, Members, Admins, Books, BookCopies,
                       Loans, Reservations, Wishlists, Notifications

    LOAD frontend routes:
        /home, /about, /credits, /contact, /catalog,
        /admin, /member, /login, /register

    SET scheduler to run NotificationCycle daily at 8:00 AM

END SYSTEM INITIALIZATION
```

---

# 2. USER AUTHENTICATION

```
FUNCTION login(email, password):
    userRecord ← QUERY Users WHERE email = email
    IF userRecord NOT FOUND:
        RETURN "Invalid email or password"

    IF HASH(password) == userRecord.passwordHash:
        CREATE sessionToken FOR userRecord.userId
        RETURN role(userRecord) AND sessionToken
    ELSE:
        RETURN "Invalid email or password"
END FUNCTION

FUNCTION logout(sessionToken):
    INVALIDATE sessionToken
END FUNCTION
```

---

# 3. ACCOUNT CREATION (GUEST → MEMBER)

```
FUNCTION createAccount(name, email, password):
    IF email ALREADY EXISTS:
        RETURN "Email already registered"

    userId ← GENERATE UUID
    INSERT INTO Users(name, email, passwordHash, role="MEMBER")

    memberId ← GENERATE UUID
    INSERT INTO Members(memberId, userId, MembershipStatus="ACTIVE",
                        maxLoans=5, phone=NULL)

    RETURN "Account created successfully"
END FUNCTION
```

---

# 4. BROWSE & SEARCH BOOK CATALOG

```
FUNCTION browseCatalog():
    RETURN ALL books ORDERED BY title
END FUNCTION

FUNCTION searchBooks(query):
    RETURN books WHERE title OR category OR author MATCH query
END FUNCTION
```

---

# 5. VIEW BOOK DETAILS

```
FUNCTION viewBookDetails(bookId):
    book ← QUERY Books WHERE bookId = bookId
    copies ← QUERY BookCopies WHERE bookId = bookId
```

```
    reservations ← COUNT Reservations WHERE bookId = bookId AND
status="in queue"
    RETURN book, copies, reservations
END FUNCTION
```

---

# 6. MEMBER WISHLIST MANAGEMENT

```
FUNCTION addToWishlist(memberId, bookId):
    IF wishlist entry EXISTS:
        RETURN "Already in wishlist"

    INSERT INTO Wishlist(memberId, bookId, createdAt=NOW)
    RETURN "Added to wishlist"
END FUNCTION
```

---

# 7. RESERVE BOOK

```
FUNCTION reserveBook(memberId, bookId):
    availableCopies ← Book.getAvailableCopies(bookId)

    IF availableCopies > 0:
        RETURN "Copy available — no need to reserve"

    queuePosition ← COUNT Reservations WHERE bookId=bookId AND
status="in queue"
    reservationId ← GENERATE UUID

    INSERT INTO Reservations(reservationId, memberId, bookId,
                             placedAt=NOW, status="in queue",
                             positionInQueue = queuePosition + 1)

    SEND Notification:
        type="confirmation"
        message="Reservation placed successfully"
```

```
    RETURN "Reservation created"
END FUNCTION
```

---

# 8. CHECK AVAILABLE COPIES (INCLUDED USE CASE)

```
FUNCTION getAvailableCopies(bookId):
    copies ← QUERY BookCopies WHERE bookId = bookId
    RETURN COUNT(status="available")
END FUNCTION
```

---

# 9. BORROW (CHECKOUT) A BOOK

```
FUNCTION borrowBook(memberId, copyId):
    IF Member has maxLoans:
        RETURN "Loan limit reached"

    IF BookCopy.status != "available":
        RETURN "Copy not available"

    loanId ← GENERATE UUID
    checkoutDate ← TODAY
    dueDate ← TODAY + 14 days

    INSERT INTO Loans(loanId, memberId, copyId,
                      checkoutDate, dueDate, returnedDate=NULL)

    UPDATE BookCopy SET status="on loan"

    RETURN "Book borrowed successfully"
END FUNCTION
```

---

# 10. RETURN BOOK

```
FUNCTION returnBook(loanId):
    loan ← QUERY Loans WHERE loanId = loanId

    UPDATE Loan SET returnedDate = TODAY
    UPDATE BookCopy SET status = "available"

    IF loan.isOverdue():
        CALCULATE overdueFee
        APPLY fee to Member account

    PROCESS reservationQueue(bookId)

    RETURN "Book returned"
END FUNCTION
```

---

# 11. OVERDUE CHECK (INCLUDED USE CASE)

```
FUNCTION isOverdue(loan):
    IF TODAY > loan.dueDate AND loan.returnedDate IS NULL:
        RETURN TRUE
    ELSE:
        RETURN FALSE
END FUNCTION
```

---

# 12. CALCULATE OVERDUE FEES (INCLUDED USE CASE)

```
FUNCTION calculateOverdueFee(loan):
    overdueDays ← (TODAY - loan.dueDate)
    feeAmount ← overdueDays * 0.50
    RETURN feeAmount
END FUNCTION
```

---

# 13. RESERVATION QUEUE HANDLING

```
FUNCTION processReservationQueue(bookId):
    nextReservation ← FIRST Reservation WHERE bookId AND status="in
queue"

    IF nextReservation EXISTS:
        UPDATE nextReservation SET status="ready"
        SEND Notification(type="ready",
recipient=nextReservation.memberId)
END FUNCTION
```

---

# 14. MEMBER VIEW LOANS & FEES

```
FUNCTION viewLoans(memberId):
    loans ← QUERY Loans WHERE memberId = memberId
    RETURN loans
END FUNCTION
```

---

# 15. RENEW MEMBERSHIP

```
FUNCTION renewMembership(memberId):
    UPDATE Member SET MembershipStatus="ACTIVE"
    RETURN "Membership renewed"
END FUNCTION
```

---

# 16. NOTIFICATION SYSTEM (TIMER / SCHEDULER)

```
DAILY NotificationCycle():
    allLoans ← QUERY Loans WHERE returnedDate IS NULL

    FOR each loan IN allLoans:
        IF loan due in 2 days:
            SEND Notification(type="due soon")

        IF loan isOverdue():
            SEND Notification(type="overdue")

END CYCLE
```

---

# 17. NOTIFICATION SENDER (INCLUDED USE CASE)

```
FUNCTION sendNotification(notification):
    GENERATE email message
    DETERMINE recipient based on memberId
    CALL EmailService.send(email)
END FUNCTION
```

---

# 18. ADMIN — MANAGE BOOKS

```
FUNCTION adminAddBook(bookData):
    bookId ← GENERATE UUID
    INSERT INTO Book(bookId, bookData...)
```

```
    FOR i ← 1 TO numberOfCopies:
        INSERT INTO BookCopy(copyId=UUID, bookId, status="available")

    RETURN "Book added"
END FUNCTION

FUNCTION adminUpdateBook(bookId, updatedFields):
    UPDATE Book SET fields = updatedFields
    RETURN "Book updated"
END FUNCTION

FUNCTION adminRemoveBook(bookId):
    DELETE FROM Book WHERE bookId = bookId
    DELETE FROM BookCopies WHERE bookId = bookId
    RETURN "Book removed"
END FUNCTION
```

---

# 19. ADMIN — MANAGE USERS

```
FUNCTION adminViewUsers():
    RETURN ALL Users WITH roles
END FUNCTION

FUNCTION adminModifyLoan(loanId, changes):
    UPDATE Loans SET fields = changes
    RETURN "Loan updated"
END FUNCTION
```

---

# 20. CONTACT FORM (EMAIL SERVICE)

```
FUNCTION submitContactForm(name, email, phone, message):
    CREATE email containing form data
    SEND email to all team members
```

```
    RETURN "Message sent"
END FUNCTION
```

---

# 21. FRONTEND PAGE FLOW / UI LOGIC

```
ROUTE /home:
    DISPLAY introduction, project goals, navigation

ROUTE /about:
    DISPLAY team members, skills, education timeline

ROUTE /credits:
    DISPLAY headshot, roles, tasks per member

ROUTE /contact:
    DISPLAY form → submitContactForm()

ROUTE /catalog:
    DISPLAY browseCatalog()

ROUTE /admin:
    REQUIRE role="ADMIN"
    DISPLAY book manager, user manager, loan dashboard

ROUTE /member:
    REQUIRE role="MEMBER"
    DISPLAY loans, wishlist, reservations, profile
```

---

# 22. SYSTEM TERMINATION

```
FUNCTION shutdownSystem():
    CLOSE database connections
    STOP notification scheduler
END FUNCTION
```