PSEUDOCODE FOR THE WHOLE PROGRAM FLOW

1. Start
2. Initialize Database Manager
3. Initialize BorrowingApp
4. Load Pending Requests
5. Main Menu Loop
    - Display Main Menu Options:
        - Borrower's Log In
        - Borrower's Registration
        - Admin Access
        - Exit
    - Handle User Input:
        - If "Borrower's Log In":
            - Call studentLogin()
        - If "Borrower's Registration":
            - Call studentRegister()
        - If "Admin Access":
            - Call adminAccess()
        - If "Exit":
            - Break Loop
6. Function: studentLogin()
    - Prompt for student name and number
    - If input is "return":
        - Return to main menu
    - Validate credentials using AccountManager
    - If valid:
        - Create BorrowingRequest
        - Call handleBorrowingRequest(request)
    - If invalid:
        - Display error message
7. Function: studentRegister()
    - Prompt for student name and number
    - If input is "return":
        - Return to main menu
    - Validate student number format
    - Register account using AccountManager
    - If successful:
        - Display success message
    - If failed:
        - Display error message
8. Function: adminAccess()
    - Prompt for admin name and password
    - If input is "return":

        - Return to main menu
    - If valid credentials:
        - Admin Menu Loop:
            - Display Admin Menu Options:
                - View Materials
                - Add Material
                - Remove Material
                - Process Borrowing Requests
                - Exit Admin Access
            - Handle User Input:
                - If "View Materials":
                    - Call viewMaterials()
                - If "Add Material":
                    - Call addMaterialAdmin()
                - If "Remove Material":
                    - Call removeMaterialAdmin()
                - If "Process Borrowing Requests":
                    - Call processAdminRequests()
                - If "Exit Admin Access":
                    - Break Loop
    - If invalid credentials:
        - Display error message
9. Function: viewMaterials()
    - Retrieve and display list of materials from dbManager
10. Function: addMaterialAdmin()
    - Call viewMaterials()
    - Prompt for material name and quantity
    - If input is "return":
        - Return to admin menu
    - Validate quantity input
    - Add material to dbManager
    - Display success message
11. Function: removeMaterialAdmin()
    - Call viewMaterials()
    - Prompt for material name
    - If input is "return":
        - Return to admin menu
    - If material exists:
        - Remove material from dbManager
        - Display success message
    - If material does not exist:
        - Display error message
12. Function: handleBorrowingRequest(request)
    - Loop until user finishes borrowing:

        - Call viewMaterials()
        - Prompt for material name and quantity
        - If input is "done":
            - Call finishBorrowing(request)
            - Break loop
        - Validate quantity input
        - Call addMaterial(request, materialName, quantity)
13. Function: addMaterial(request, materialName, quantity)
    - Check if material exists in dbManager
    - If exists:
        - Add material to request
        - Deduct quantity from dbManager
        - Display success message
    - If not:
        - Display error message
14. Function: processAdminRequests()
    - Display number of pending requests
    - If there are pending requests:
        - Process the first request in the queue
        - Display requested materials
        - Prompt for approval decision
        - Log the request and decision
        - If approved:
            - Display approval message
        - If denied:
            - Return materials to dbManager
            - Display denial message
        - Save updated queue to file
    - If no pending requests:
        - Display message
15. Function: finishBorrowing(request)
    - If no materials requested:
        - Display message
    - If materials requested:
        - Add request to queue
        - Log the request
        - Display submission message
16. End

PSEUDOCODE FOR EACH CLASS (For Procedure Part as Guide for Coding)

- **Material Class**

```
Class Material
   Properties
      String name
      Integer quantity

   Constructor Material(name, quantity)
      this.name = name
      this.quantity = quantity
End Class
```

- **Account Manager Class**

```
Class AccountManager
   Properties
      String filename

   Constructor AccountManager(filename = "accounts.txt")
      this.filename = filename

   Method registerAccount(name, studentNumber)
      Open file for reading filename
      While reading each line from file
         If existingName == name OR existingNumber == studentNumber
            Return false  // User already exists
      End While

      Open file for appending filename
      Write name, studentNumber to file
      Return true  // Registration successful

   Method login(name, studentNumber)
      Open file for reading filename
      While reading each line from file
         If existingName == name AND existingNumber == studentNumber
            Return true  // Login successful
      End While
      Return false  // Credentials incorrect
```

```
End Class
```

- **Database Manager Class**

```
Class DatabaseManager
    Properties
        String filename
        Map<String, Integer> materials

    Constructor DatabaseManager(filename = "database.txt")
        this.filename = filename
        Call loadMaterials()

    Method loadMaterials()
        Open file for reading filename
        While reading each line from file
            Parse name and quantity
            Store name with quantity in materials map

    Method saveMaterials()
        Open file for writing filename
        For each material in materials
            Write material.name, material.quantity to file

    Method addMaterial(name, quantity)
        Increment materials[name] by quantity
        Call saveMaterials()

    Method removeMaterial(name)
        Remove name from materials
        Call saveMaterials()

    Method getMaterials()
        Return materials
End Class
```

- **Borrowing Request Class**

```
Class BorrowingRequest
    Properties
```

```
    String studentName
    String studentNumber
    List<Material> requestedMaterials

  Constructor BorrowingRequest(name, number)
    this.studentName = name
    this.studentNumber = number
End Class
```

- **Borrowing App Class**

```
Class BorrowingApp
  Properties
    DatabaseManager dbManager
    Queue<BorrowingRequest> borrowingQueue
    LogFile logFile
    String pendingRequestsFile = "pendingRequests.txt"

  Constructor BorrowingApp()
    Call loadPendingRequests()

  Destructor ~BorrowingApp()
    Call savePendingRequests()

  Method loadPendingRequests()
    Open pendingRequestsFile for reading
    While reading each line from file
      Create BorrowingRequest from line data
      Push request to borrowingQueue

  Method savePendingRequests()
    Open pendingRequestsFile for writing
    While borrowingQueue is not empty
      Write request data to file

  Method start()
```

```
    While true
        Display main menu
        Read choice
        Execute corresponding method (studentLogin, studentRegister, adminAccess, exit)

Method studentLogin()
    Read student name and number
    If valid credentials
        Create BorrowingRequest
        Call handleBorrowingRequest(request)

Method studentRegister()
    Read student name and number
    Validate student number
    Call AccountManager.registerAccount()

Method adminAccess()
    Authenticate admin
    Display admin menu
    Execute corresponding method based on choice

Method handleBorrowingRequest(request)
    While true
        Display available materials
        Read material name and quantity
        If "done"
            Call finishBorrowing(request)
            Break
        Call addMaterial(request, materialName, quantity)

Method addMaterial(request, materialName, quantity)
    Check availability in dbManager
    Add to request and update dbManager materials

Method processAdminRequests()
    While borrowingQueue is not empty
        Process top request
        Prompt admin for decision
        Log decision and save changes

Method finishBorrowing(request)
    If no materials requested
        Inform user
    Add request to borrowingQueue
```

    Log borrowing submission
End Class

- **Main Function**

Function main()
  Create instance of BorrowingApp
  Call start() on app instance
End Function