Conceptual Model

Objective:

Use your requirements document to develop a comprehensive Entity-Relationship (ER) model that accurately represents the data requirements and relationships for your database project. This model will serve as a blueprint for your database design, capturing the essential entities, relationships, and constraints based on the requirements you have gathered. No formal template is required, but the following sections should be included in a requirement document (the *italic* parts are subsections).

Intro:

Purpose

We are developing a Library Management System to maintain a database for a local library, allowing members of the community to effectively and efficiently use library resources. By implementing this system, library workers can track important information on user activity, while also being able to adjust the rules to their needs (like changing membership status), generate reports, etc. In general, this system will provide ease to the community, the staff, and help make people's experience at the library more enjoyable.

Scope

Our Library Management System efficiently tracks books, digital media, and magazines, maintaining key details like title, author, ISBN, publication year, genre, and availability. It also monitors guest desktop computers, recording user sessions and login/logout times. The system manages item availability, loan statuses, due dates, and check-out history while allowing users to search the catalog, reserve and renew items, and check loan statuses. Library staff can process checkouts, returns, add new items, and manage user accounts, which store information on checked-out items and holds. Additionally, the system will only generate reports on borrowing trends by genre.

Entities & Attributes:

Identify Entities: List all the major entities that will be part of your database. This includes the initial entities in the project description, the ones you identified during the requirements engineering, and the additional ones during your team brainstorming.

Define Attributes: For each entity, list its attributes and specify the data types or constraints. For example, the `Book` entity might have attributes such as `ISBN`, `Title`, `Author`, `Genre, Price,` and `Stock Quantity`.

Data Entities:

- User
    - Contact Information
        - FirstName
        - LastName
        - PhoneNumber
        - Email
        - Address
            - Zipcode
    - Specialization-Library Member (View permissions)
        - UserID
        - MembershipStatus
        - ItemsCheckedOut
        - DateJoined
        - OutsandingBalance
        - Sex
        - DOB
    - Specialization-Library Staff (View/Edit permissions)
        - StaffID
        - Salary
        - PositionName
        - PositionType (ie fulltime, part-time, volunteer)
    - Specialization-System Administrator (View/Edit permissions)
        - StaffID
            - SecurityKey
        - Managees

- Item
    - ItemID
    - ISBN

    - Title
    - Author
    - PublicationYear
    - Publisher
    - Genre
    - AvailabilityStatus
    - Quantity
        - How to track???
    - Specialization-Book
        - QuantityAvailable
        - DueDate
    - Specialization-Digital Media
        - Medium

- Copy
    - ItemID
    - ISBN
    - Title
    - Quantity
    - QuantityAvailable

- Transaction
    - CheckedOutID
    - UserID
    - ItemID
    - Status
    - DateCheckedOut
    - DueDate
    - NumRenewals

- Report
    - ReportID
    - BorrowingTrends
    - NumberofCheckouts
    - Date
    - TimePeriod

- Hold
    - ItemID
    - UserID

    - HoldID
    - Date

- Fine
    - UserID
    - ItemID
    - Amount

Relationships:

Define Relationships: Determine how the entities are related to each other. Define the multiplicity (one-to-one, one-to-many, many-to-many) and any constraints. For example, a `Book` can be written by one or more `Author`, and a `Purchase` can include multiple `Book`.

- User → is a
  - Library Member → rent item | return item | holds Item | pays OutstandingBalance | check OutstandingBalance | requests renewal of Item | incurs fine for overdue Transaction
  - Library Staff → add new item | delete item | add Member | Remove Member | Suspend Member | Generate report
  - System Administrator → grant access to new Library Member(read)/Library Staff(read/write) | remove access to new Library Member(read)/Library Staff(read/write) | manage Staff Member
- Item → is a
  - Book → has quantity | rented by Library Member | added to inventory | deleted from inventory | incur fine | held by Library Member | renewed by Library Member
  - Digital Media has quantity | rented by Library Member | added to inventory | deleted from inventory | incur fine | held by Library Member | renewed by Library Member| is medium
- Transaction → records
  - rental of item by Library Member
  - return of item by Library Member
  - fine incurred for overdue item
  - fine payment by Library Member
  - renewal request for item by Library Member
  - hold request for item by Library Member
- Report → record
  - Generate for SysAdmin
  - Viewed by LibraryStaff | SysAdmin
  - Include BorrowingTrends | NumberofCheckouts | Date | Time Period
  - Tracks OverdueItems | PopularGenres | FrequentBorrowers | OutstandingBalances
  - Exported as PDF | CSV | Excel
  - Scheduled for daily | weekly | monthly | custom
- Hold → records
  - hold request placed by Library Member for Item
  - held by Library Member

    - expires after a set period
    - Fulfilled by Library Member when Item is checked out
    - cancelled by Library Member
    - removed when expired or checked out
    - Is linked to Item
- Inventory
    - Item is recorded in inventory
    - Library_staff updates inventory
    - Transaction affects inventory
-