Database Systems Design Phase 1 Report

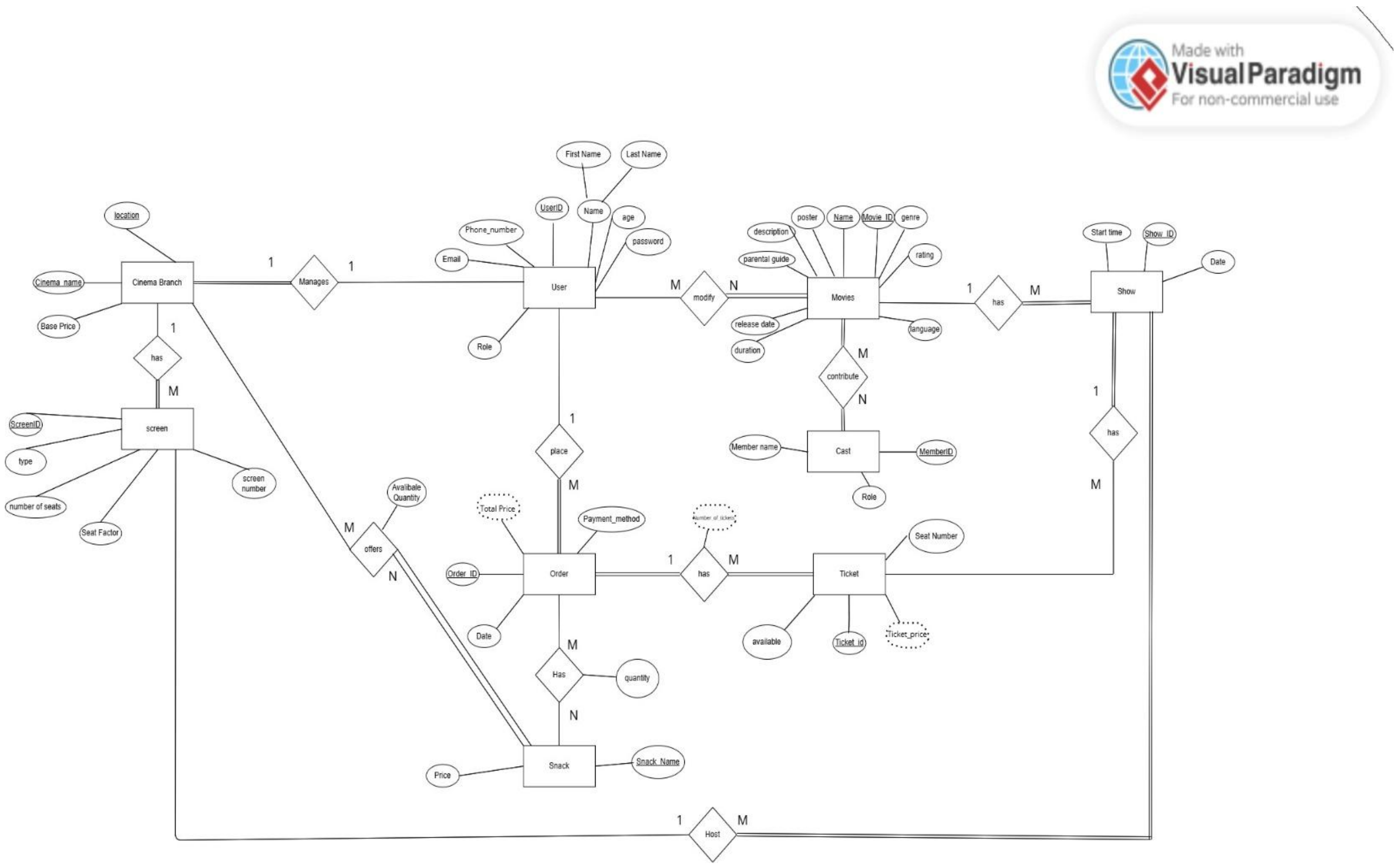
CINEMA BOOKING SYSTEM

Team Number:2
Adham Walid Said 23P0024
Eslam Mohamed Fawzy 23p0052
Carol Kamal Magdy 23P0328
Hassan Ismail 23p0152
Loay Mahmoud 23p0419
Toka Elsayed 23P0044
TA: Eng Esraa Karam

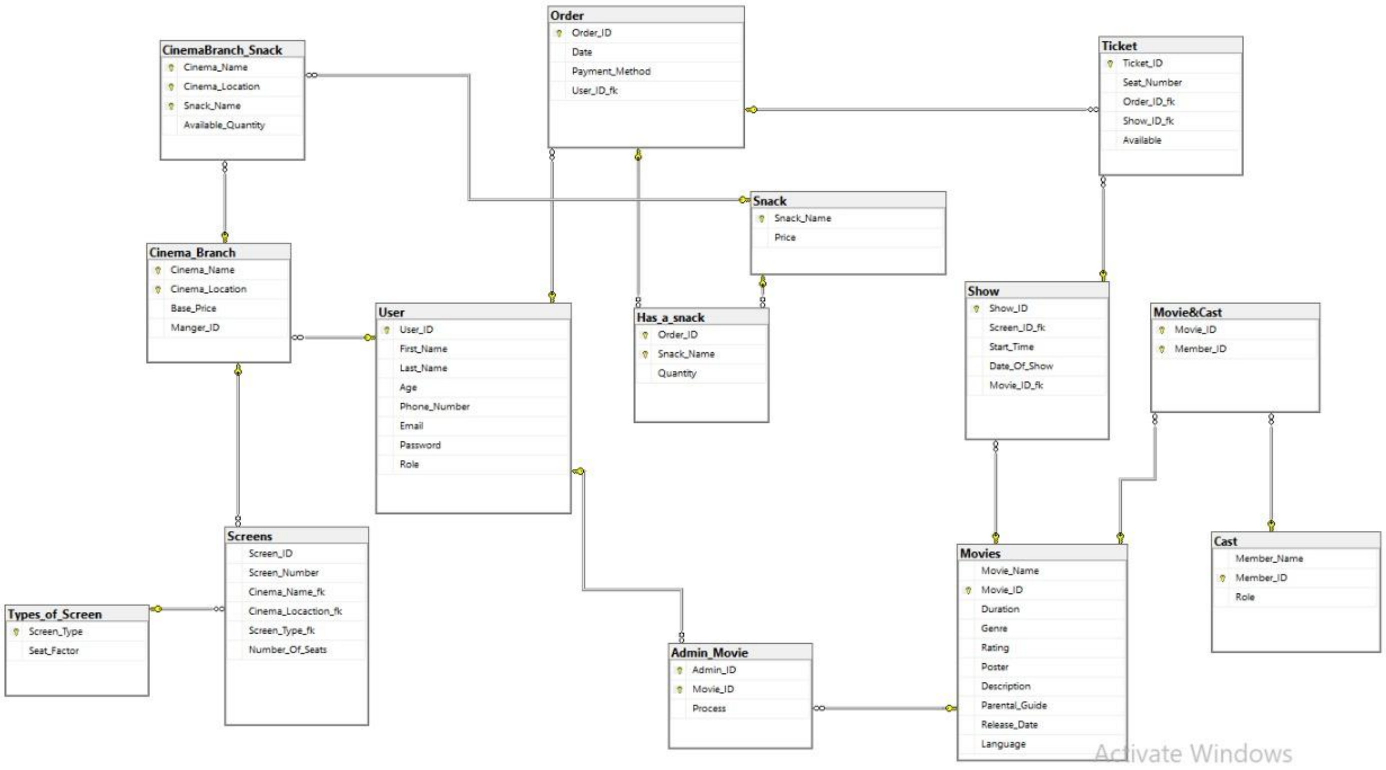
Introduction:

Our Cinema Booking System is designed to manage the entire movie booking process with options to book in different cinemas and in different Branches, also variety of processes from adding movies to purchasing tickets. It handles movie scheduling, user interactions, and ticket reservations, creating a mini-world that covers all aspects of cinema management. Using tables and diagrams, we structured the system to efficiently store and manage data related to users, movies, shows, tickets, and orders. This phase focuses on designing a database that organizes and connects these elements seamlessly, ensuring smooth operations and an intuitive booking experience.

1-Entity-Relationship Diagram



2-Schema


CinemaBranch_Snack
Cinema_Name
Cinema_Location
Snack_Name
Available_Quantity
Order
Order_ID
Date
Payment_Method
User_ID_fk
Ticket
Ticket_ID
Seat_Number
Order_ID_fk
Show_ID_fk
Available
Snack
Snack_Name
Price
Cinema_Branch
Cinema_Name
Cinema_Location
Base_Price
Manger_ID
User
User_ID
First_Name
Last_Name
Age
Phone_Number
Email
Password
Role
Has_a_snack
Order_ID
Snack_Name
Quantity
Show
Show_ID
Screen_ID_fk
Start_Time
Date_Of_Show
Movie_ID_fk
Movie&Cast
Movie_ID
Member_ID
Screens
Screen_ID
Screen_Number
Cinema_Name_fk
Cinema_Locaction_fk
Screen_Type_fk
Number_Of_Seats
Types_of_Screen
Screen_Type
Seat_Factor
Admin_Movie
Admin_ID
Movie_ID
Process
Movies
Movie_Name
Movie_ID
Duration
Genre
Rating
Poster
Description
Parental_Guide
Release_Date
Language
Cast
Member_Name
Member_ID
Role
Activate Windows

3-Design

Idea and Choices:

1 - Cinema_Branch

- Represents different cinema locations, ensuring each theater is uniquely identified.
- Attributes like **Cinema_Name** and **Cinema_Location** prevent conflicts when managing multiple branches.

2 - Screens

- Each cinema consists of multiple screens where movies are shown.
- Storing **Screen_Type** and **Number_Of_Seats** helps in ticket pricing and seat availability calculations.

3 - User

- Covers customers, admins, and managers.
- The **Role** attribute differentiates between regular users (booking tickets), managers (handling cinemas), and admins (managing movies).

4 - Order

- Tracks ticket purchases, ensuring each order is linked to a specific user.
- Attributes like **Total_Price** and **Payment_Method** allow seamless payment processing.

5 - Snack

- Allows customers to purchase food alongside their tickets.
- Necessary for tracking **inventory** and **pricing** within the cinema.

6 - Ticket

- Represents individual show tickets, ensuring seat allocation and availability.
- **Seat_Number** and **Availability** attributes prevent double-booking.

7 - Show

- Each show represents a scheduled movie screening with a specific **start time** and **date**.
- Essential for managing which movie is playing on which screen.

8 - Movie

- Central to the system, storing details such as **Genre, Rating, and Release_Date**.
- Used for filtering movies based on user preferences**.**

9 - Cast

- Stores actor and director details for each movie.
- Enables users to browse films based on their favorite actors.

Entities attributes:

1. Cinema Branch
- Cinema_Name(Part of Composite Key)
- Cinema_Location(Part of Composite Key)
- Base_Price

2. Screens
- Screen_ID(PK)
- Screen_Number
- Screen_Type
- Number_Of_Seats
- Seat_Factor

3. User
- User_ID(PK)
- Name(Composes of First Name ,Last Name)
- Age
- Phone_Number
- Email
- Password
- Role

4. Order
- Order_ID (PK)
- Date
- Total_Price
- Payment_Method

5. Snack
- Snack_Name (PK)
- Price

6. Ticket
- Ticket_ID (PK)
- Seat_Number
- Available

7. Show
- Show_ID (PK)
- Start_Time
- Date_Of_Show

8. Movies
- Movie_ID (PK)
- Movie_Name
- Duration
- Genre
- Rating
- Poster
- Description
- Parental_Guide
- Release_Date
- Language

9. Cast
- Member_ID (PK)
- Member_Name
- Role

Entities in the system represent real-world components of the cinema booking process. They include **Cinema**, **Screens**, and **Show** for theater management, **User** for customer interactions, **Movies** and **Cast** for film details, and **Order**, **Ticket**, and **Payment_Method** for booking and transactions. Additionally, **Snack** allows for food purchases, enhancing the user experience.

Relationships:

1- User Places Order

- **1:M relationship** – A user can place multiple orders, but each order belongs to only one user.
- **Foreign Key:** Order(User_ID) → User(User_ID) (User_ID in Order references User table).

2- User Manages Cinema

- **1:1 relationship** – A manager is responsible for one cinema, and each cinema has one manager.
- **Foreign Key:** Cinema(Manager_ID) → User(User_ID) (Only users with a "Manager" role).

3- User Administers Movies

- **M:N relationship** – Multiple admins can modify multiple movies.
- **Junction Table:** Admin_Movie(User_ID, Movie_ID), where
  - Admin_Movie(User_ID) → User(User_ID)
  - Admin_Movie(Movie_ID) → Movie(Movie_ID)

4- Cinema Has Screens

- **1:M relationship** – A cinema has multiple screens, but each screen belongs to one cinema.
- **Foreign Key:** Screen(Cinema_Name, Cinema_Location) → Cinema(Cinema_Name, Cinema_Location).

5- Screen Hosts Shows

- **1:M relationship** – A screen can have multiple shows, but each show takes place on only one screen.
- **Foreign Key:** Show(Screen_ID) → Screen(Screen_ID).

6- Show Features Movie

- **1:M relationship** – Multiple shows can feature the same movie, but each show plays only one movie.
- **Foreign Key:** Show(Movie_ID) → Movie(Movie_ID).

7- Order Contains Tickets

- **1:M relationship** – An order can include multiple tickets, but each ticket belongs to one order.
- **Foreign Key:** Ticket(Order_ID) → Order(Order_ID).

8- Ticket Belongs to Show

- **1:M relationship** – Multiple tickets can be issued for the same show, but each ticket is linked to only one show.
- **Foreign Key:** Ticket(Show_ID) → Show(Show_ID).

9- Order Includes Snacks

- **M:N relationship** – An order can contain multiple snacks, and each snack can be included in multiple orders.
- **Junction Table:** Order_Snack(Order_ID, Snack_Name), where
  - Order_Snack(Order_ID) → Order(Order_ID).
  - Order_Snack(Snack_Name) → Snack(Snack_Name).

10- Movie Has Cast Members

- **M:N relationship** – A movie can have multiple cast members, and each cast member can appear in multiple movies.
- **Junction Table:** Movie_Cast(Movie_ID, Member_ID), where
  - Movie_Cast(Movie_ID) → Movie(Movie_ID).
  - Movie_Cast(Member_ID) → Cast(Member_ID).

11- Cinema Offers Snacks

- **M:N relationship** – Every Cinema_branch can Offer Multiple Types of snacks, and each snack can be offered by many cinema branches
- **Junction Table:** CinemaBranch_Snack(CinemaName,CinemaLocation,SnackName, Quantity), where
  - CinemaBranch_Snack (Cinema Name,Cinema Location) → Cinema_Branch(Cinema Name,Cinema Location).
  - CinemaBranch_Snack (Snack Name) → Snacks(Snack Name).

Our cinema booking system connects all tables using **foreign keys** to keep data organized and accurate. Users can place orders, cinemas have screens, and shows feature movies. Some relationships, like **movies with cast members** or **orders with snacks**, use **junction tables** to handle many-to-many connections. This setup ensures smooth operations and prevents data errors.

4-Normalization:

Our database design carefully applies **normalization principles (1NF, 2NF, and 3NF)** to ensure **data integrity, reduce redundancy, and optimize performance**. Below is how each level of normalization was handled:

First Normal Form (1NF) – Eliminating Repeating Groups

- Each attribute contains **atomic (indivisible) values**.
- For example, in the **User** table, we ensure that attributes like **Phone_Number** and **Email** hold only **single values per row**, rather than multiple entries in one field.

Second Normal Form (2NF) – Removing Partial Dependencies

- Every **non-key attribute** is **fully dependent** on the **entire primary key**.
- In the **Order** table, attributes like **Total_Price** and **Payment_Method** depend **only on Order_ID**, ensuring that no field is dependent on just a part of a composite key.
- Many-to-Many relationships, such as **Movies and Cast Members**, are correctly handled using **junction tables** (e.g., Movie_Cast(Movie_ID, Member_ID)), preventing partial dependencies.

Third Normal Form (3NF) – Eliminating Transitive Dependencies

initially, the **screens** table contained both **screen_type** and **seat_factor**, creating a **transitive dependency** (screen_type → seat_factor), which violates **3NF**.

To resolve this, **a new table (screen_types) was introduced**, storing screen_type and its corresponding seat_factor. The screens table now references screen_type as a **foreign key**, ensuring **3NF compliance** by eliminating transitive dependency and improving data integrity.

These Are only Examples of how Normalization Was handled with all tables,Each and Every Other Table Follows the Three Forms of Normalization