

UNOX Movie Experience Platform

1. Introduction

The UNOX Movie Experience Platform is designed to modernize and digitize the end-to-end process of movie booking, ticketing, food ordering, and loyalty management for a multiplex or cinema chain. This case study allows students to apply their understanding of data modelling, relational database design, SQL operations, and analytics within a realistic business context.

The goal is to simulate a **real-world cinema management system** that integrates multiple workflows — ticket booking, seat allocation, payments, membership rewards, and food delivery — into a unified digital experience.

2. Business Overview









Traditional cinema operations often involve fragmented systems for ticketing, food orders, and customer loyalty. UNOX aims to consolidate these into a single intelligent platform that enhances the movie-going experience while enabling better operational efficiency and data-driven decision-making.

3. Objectives

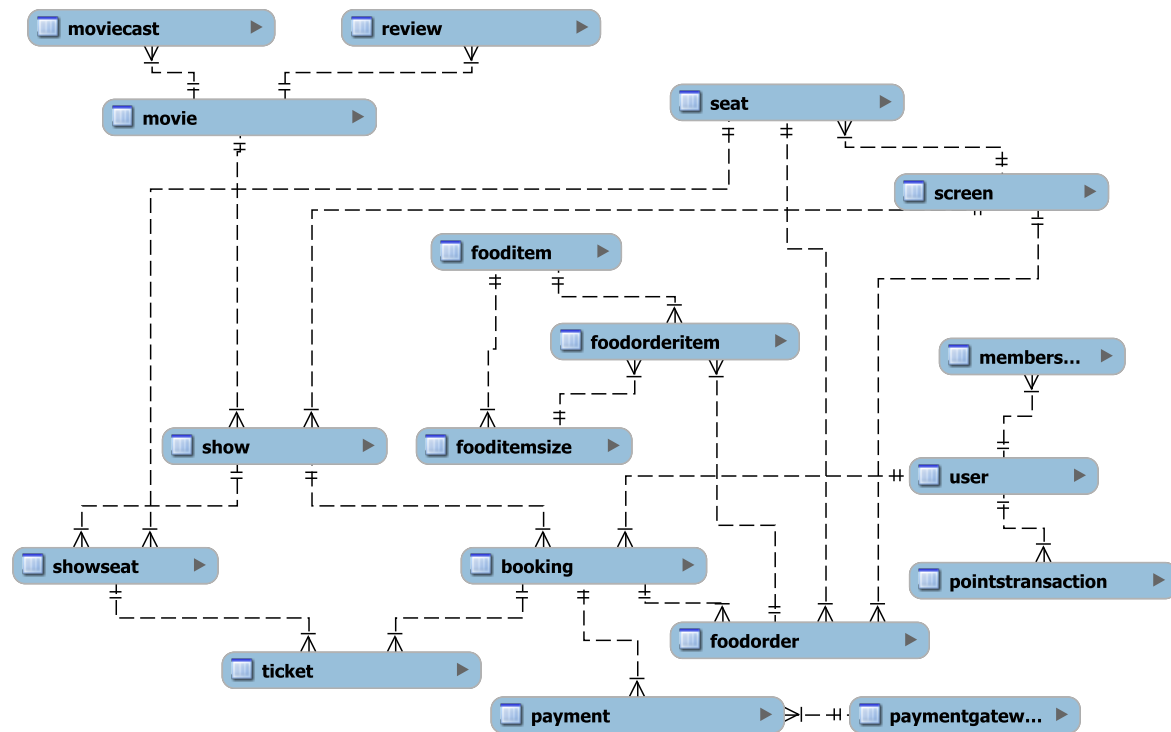
- Simplify booking and payment processes for users.
- Manage screens, shows, and seat availability efficiently.
- Automate food ordering linked to seats and bookings.
- Introduce loyalty programs and points-based rewards.
- Enable management insights through reviews, payments, and usage data.

4. Scope of the System

The **UNOX platform** includes the following core components:

Module	Description
 Movie Management	Handles movie details, cast, reviews, and status.
 Screen & Seat Management	Defines cinema screens, their capacities, and seat layouts.
 Show Scheduling	Manages show timings, linked movies, and screen allocation.
 Booking & Ticketing	Allows users to book seats, receive QR-coded tickets, and track downloads/scans.
 Payments	Supports multi-gateway payments, transaction tracking, and failure reasons.
 Food Ordering System	Enables pre-ordering and in-seat delivery of snacks or combos linked to bookings.
 Membership & Points	Tracks user loyalty, points accumulation, and transaction history.
 Reviews & Ratings	Collects viewer feedback to improve content curation and marketing.

5. Database, Tables & Entity Relationships



The **UNOX Database Schema** captures all the necessary entities and their relationships across modules. Each entity (e.g., User, Booking, Show, Payment, FoodOrder) is interlinked through **primary and foreign keys** to maintain referential integrity and enable complex business analytics.

A. Master Tables

These store **static or foundational data** that defines the structure of the business. They rarely change and are referenced by many other tables.

Table	Description
Screen	Defines movie halls in the multiplex.
Seat	Seat layout for each screen.
Movie	Catalog of all movies.
MovieCast	List of actors, directors, etc., for each movie.
FoodItem	Food menu items or combos.
FoodItemSize	Size and price variants for food items.
PaymentGateway	Reference list of supported payment providers.
User	Registered users (acts as a customer master).

B. Transactional Tables

These store dynamic, time-bound records — business events that occur during daily operations. Each record represents a specific transaction or user activity.

Table	Description
Booking	Records each booking made by a user.
Ticket	Stores generated tickets linked to a booking.
Payment	Logs details of each payment transaction.
FoodOrder	Food orders placed for a specific booking.
FoodOrderItem	Line items (each food item ordered) in a food order.
PointsTransaction	Loyalty or reward point transactions.
Review	User reviews and ratings for movies.

C. Operational (or Process) Tables

These handle runtime operations that link master and transactional data — they represent the business workflow or scheduling logic of the system.

Table	Description
Show	Defines which movie is shown on which screen and when.
ShowSeat	Tracks seat availability per show.
Membership	Links users to their loyalty program status and current points.

5. Expected Learning Outcomes

- By working on this case study, students will:
- Understand **relational database design** and normalization principles.
- Practice **SQL DDL and DML** operations such as CREATE, INSERT, SELECT, JOIN, and GROUP BY.
- Write **queries for analytics**, such as total sales, occupancy, and customer loyalty.
- Implement **foreign key relationships** and constraints in SQL.
- Understand how to apply ETL.
- Generate Actionable Business Insights and Visualization.
- Learn to interpret **real-world business data models** and translate them into database logic.

6. Future Enhancements (For Advanced Learners)

- Implement dynamic pricing based on seat class and demand.
- Add coupon code and discount management.
- Build recommendation systems for movies and combos.
- Integrate with third-party APIs for payment or reviews.
- Enable analytics dashboards using Power BI / Tableau / Python.

7. UNOX Database Schema

1. Screen

Column	Type	Constraints
screen_id	INT	PK, Auto Increment
name	VARCHAR(50)	NOT NULL
class_type	VARCHAR(10)	NOT NULL
capacity	INT	NOT NULL

2. Seat

Column	Type	Constraints
seat_id	INT	PK, Auto Increment
screen_id	INT	FK → Screen(screen_id)
seat_number	VARCHAR(10)	NOT NULL

3. Movie

Column	Type	Constraints
movie_id	INT	PK, Auto Increment
title	VARCHAR(255)	NOT NULL
genre	VARCHAR(50)	NOT NULL
rating	DECIMAL(3,1)	NOT NULL
status	VARCHAR(20)	NOT NULL
poster_image_url	VARCHAR(255)	NULL

4. MovieCast

Column	Type	Constraints
cast_id	INT	PK, Auto Increment
movie_id	INT	FK → Movie(movie_id)
person_name	VARCHAR(100)	NOT NULL
role	VARCHAR(100)	NOT NULL

5. Review

Column	Type	Constraints
review_id	INT	PK, Auto Increment
movie_id	INT	FK → Movie(movie_id)
content	TEXT	NOT NULL
review_date	DATETIME	NOT NULL
reviewer_name	VARCHAR(100)	NOT NULL
rating	INT	

6. Show

Column	Type	Constraints
show_id	INT	PK, Auto Increment
screen_id	INT	FK → Screen(screen_id)
movie_id	INT	FK → Movie(movie_id)
show_datetime	DATETIME	NOT NULL

7. ShowSeat

Column	Type	Constraints
show_seat_id	INT	PK, Auto Increment
show_id	INT	FK → Show(show_id)
seat_id	INT	FK → Seat(seat_id)
is_available	BOOLEAN	NOT NULL, DEFAULT TRUE

8. User

Column	Type	Constraints
user_id	INT	PK, Auto Increment
name	VARCHAR(100)	NOT NULL
email	VARCHAR(150)	NOT NULL
phone	VARCHAR(15)	NULL

9. Membership

Column	Type	Constraints
membership_id	INT	PK, Auto Increment
user_id	INT	FK → User(user_id)
current_points	INT	NOT NULL, DEFAULT 0

10. Booking

Column	Type	Constraints
booking_id	INT	PK, Auto Increment
user_id	INT	FK → User(user_id)
show_id	INT	FK → Show(show_id)
booking_datetime	DATETIME	NOT NULL
total_cost	DECIMAL(10,2)	NOT NULL

11. Ticket

Column	Type	Constraints
ticket_id	INT	PK, Auto Increment
booking_id	INT	FK → Booking(booking_id)
show_seat_id	INT	FK → ShowSeat(show_seat_id)
qr_code	VARCHAR(100)	NOT NULL
delivery_method	VARCHAR(50)	NOT NULL
is_downloaded	BOOLEAN	NOT NULL, DEFAULT FALSE
scanned_at	DATETIME	NULL

12. PaymentGateway

Column	Type	Constraints
gateway_id	INT	PK, Auto Increment
name	VARCHAR(100)	NOT NULL

13. Payment

Column	Type	Constraints
payment_id	INT	PK, Auto Increment
booking_id	INT	FK → Booking(booking_id)
gateway_id	INT	FK → PaymentGateway(gateway_id)
transaction_amount	DECIMAL(10,2)	NOT NULL
transaction_datetime	DATETIME	NOT NULL
status	VARCHAR(20)	NOT NULL
failure_reason	TEXT	NULL
credit_card_name	VARCHAR(100)	NULL
credit_card_number	VARCHAR(20)	NULL
expiry_date	DATE	NULL
cvv	VARCHAR(4)	NULL

14. FoodItem

Column	Type	Constraints
item_id	INT	PK, Auto Increment
name	VARCHAR(100)	NOT NULL
description	TEXT	NULL
is_combo	BOOLEAN	NOT NULL, DEFAULT FALSE

15. FoodItemSize

Column	Type	Constraints
size_id	INT	PK, Auto Increment
item_id	INT	FK → FoodItem(item_id)
size_name	VARCHAR(50)	NOT NULL
rate	DECIMAL(10,2)	NOT NULL

16. FoodOrder

Column	Type	Constraints
order_id	INT	PK, Auto Increment
booking_id	INT	FK → Booking(booking_id)
screen_id	INT	FK → Screen(screen_id)
seat_id	INT	FK → Seat(seat_id)
order_datetime	DATETIME	NOT NULL
total_cost	DECIMAL(10,2)	NOT NULL
delivery_method	VARCHAR(50)	NOT NULL

17. FoodOrderItem

Column	Type	Constraints
order_item_id	INT	PK, Auto Increment
order_id	INT	FK → FoodOrder(order_id)
item_id	INT	FK → FoodItem(item_id)
size_id	INT	FK → FoodItemSize(size_id)
quantity	INT	NOT NULL
price_at_time	DECIMAL(10,2)	NOT NULL

18. PointsTransaction

Column	Type	Constraints
transaction_id	INT	PK, Auto Increment
user_id	INT	FK → User(user_id)
amount	DECIMAL(10,2)	NOT NULL
points_earned	INT	NOT NULL
transaction_datetime	DATETIME	NOT NULL
transaction_type	VARCHAR(20)	NOT NULL

8. Relationships

One-to-One (1:1) Relationships

These are rare and indicate that each record in one table corresponds to exactly one record in another.

Relationship	Type	Explanation
User → Membership	1:1	Each user has at most one membership record (loyalty account).
Booking → Payment	1:1 (conceptually)	Typically, one booking corresponds to one payment record , though systems can allow partial or split payments.

One-to-Many (1:N) Relationships

These form the core of the schema, linking masters to their dependent or transactional records.

Parent Table	Child Table	Relationship	Description
Screen	Seat	1:N	One screen has many seats.
Screen	Show	1:N	One screen can host many shows.
Movie	Show	1:N	One movie can be shown multiple times.
Movie	Review	1:N	One movie can have many reviews.
Movie	MovieCast	1:N	One movie can have many cast members.
Show	ShowSeat	1:N	One show has many seats available.

Seat	ShowSeat	1:N	A seat can appear in many shows (mapped per show).
User	Booking	1:N	A user can make multiple bookings.
Show	Booking	1:N	One show can have many bookings.
Booking	Ticket	1:N	One booking can generate multiple tickets.
Booking	Payment	1:N	One booking can have multiple payment attempts (if allowed).
Booking	FoodOrder	1:N	One booking can have multiple food orders.
FoodOrder	FoodOrderItem	1:N	One food order can have multiple items.
FoodItem	FoodItemSize	1:N	One food item has multiple size/price variants.
User	PointsTransaction	1:N	One user can have multiple points transactions.
User	Membership	1:1 or 1:N	Typically one membership per user.
PaymentGateway	Payment	1:N	Each payment gateway handles multiple transactions.

Many-to-Many (M:N) Relationships

In relational databases, these are usually implemented using link tables (junction tables) that break M:N into two 1:N relationships.

Relationship	Link / Junction Table	Explanation
Show ↔ Seat	ShowSeat	Each show has many seats, and each seat can be reused in many shows.
Booking ↔ ShowSeat	Ticket	A booking can have multiple show seats, and each show seat can belong to different bookings over time.
Booking ↔ FoodItem	FoodOrderItem	A booking (via food order) can include multiple food items; a food item can appear in many orders.
User ↔ PointsTransaction	<i>(direct relationship)</i>	Each user earns points through multiple transactions, but each transaction belongs to one user (technically 1:N, but behaviorally M:N when including booking/payment linkages).