# Domain Class Diagram for AcmePlex Ticket Reservation System

## 1. Key Classes and Their Stereotypes

The following key classes and their stereotypes are identified:  
- `User` – <<Entity>>  
- `Movie` – <<Entity>>  
- `Theater` – <<Entity>>  
- `Showtime` – <<Entity>>  
- `Seat` – <<Entity>>  
- `Ticket` – <<Entity>>  
- `Payment` – <<Entity>>  
- `ContactForm` – <<Entity>>

## 2. Class Relationships

Relationships between the classes include:  
- `User ↔ Ticket`: 1-to-many (A user can purchase multiple tickets).  
- `Movie ↔ Showtime`: 1-to-many (A movie can have multiple showtimes).  
- `Showtime ↔ Seat`: 1-to-many (A showtime contains multiple seats).  
- `Theater ↔ Showtime`: 1-to-many (A theater hosts multiple showtimes).  
- `User ↔ Payment`: 1-to-many (A user can make multiple payments).  
- `Ticket ↔ Seat`: 1-to-1 (A ticket is linked to a single seat).  
- `ContactForm ↔ User`: 1-to-1 (A user can submit a single query at a time).

## 3. Multiplicities Between Classes

The following multiplicities are defined:  
- `User` ↔ `Ticket`: 1-to-many  
- `Movie` ↔ `Showtime`: 1-to-many  
- `Showtime` ↔ `Seat`: 1-to-many  
- `Theater` ↔ `Showtime`: 1-to-many  
- `User` ↔ `Payment`: 1-to-many  
- `Ticket` ↔ `Seat`: 1-to-1

## 4. Attributes for Classes

Attributes for each class:  
- `User`: userID, address, email, username, password, cardNum, cvvHash  
- `Movie`: movieID, title, rating, theaterId, releaseDate  
- `Theater`: theaterId, theaterName  
- `Showtime`: showtimeId, movieId, theaterId, time, availableSeats  
- `Seat`: seatId, isAvailable, rowNum, seatNum  
- `Ticket`: ticketID, price, seatNumber, userID  
- `Payment`: paymentID, createdAt, email, movie, paymentStatus, selectedSeats, showTime, totalPrice  
- `ContactForm`: formID, userName, email, message

## 5. Operations for Classes

Operations for each class:  
- `User`: login(), register(), searchMovie()  
- `Movie`: getShowtimes(), calculateAverageRating()  
- `Theater`: listMovies(), getShowtimes()  
- `Showtime`: getAvailableSeats(), assignSeats(selectedSeats: List<int>)  
- `Seat`: reserveSeat(), releaseSeat()  
- `Ticket`: generateTicket(), cancelTicket()  
- `Payment`: processPayment(), generateReceipt()  
- `ContactForm`: submitQuery()

## 6. Instructions for Drawing the Diagram in Draw.io

1. \*\*Add Classes\*\*:  
 - Drag and drop rectangles for each class.  
 - Divide the rectangles into three sections:  
 - Top Section: Class name with stereotype (e.g., <<Entity>> User).  
 - Middle Section: Attributes.  
 - Bottom Section: Operations.  
  
2. \*\*Connect Classes\*\*:  
 - Use simple lines for associations.  
 - Use open diamonds for aggregations (e.g., Theater ↔ Showtime).  
 - Use filled diamonds for compositions (e.g., Showtime ↔ Seat).  
  
3. \*\*Add Multiplicities\*\*:  
 - Label the connections:  
 - 1..\* for one-to-many.  
 - 1..1 for one-to-one.  
  
4. \*\*Style the Diagram\*\*:  
 - Use colors for better readability.  
 - Align and group related classes for clarity.

## 7. Save and Export the Diagram

1. Save your diagram in Draw.io format for future edits.  
2. Export as PNG or PDF for sharing.