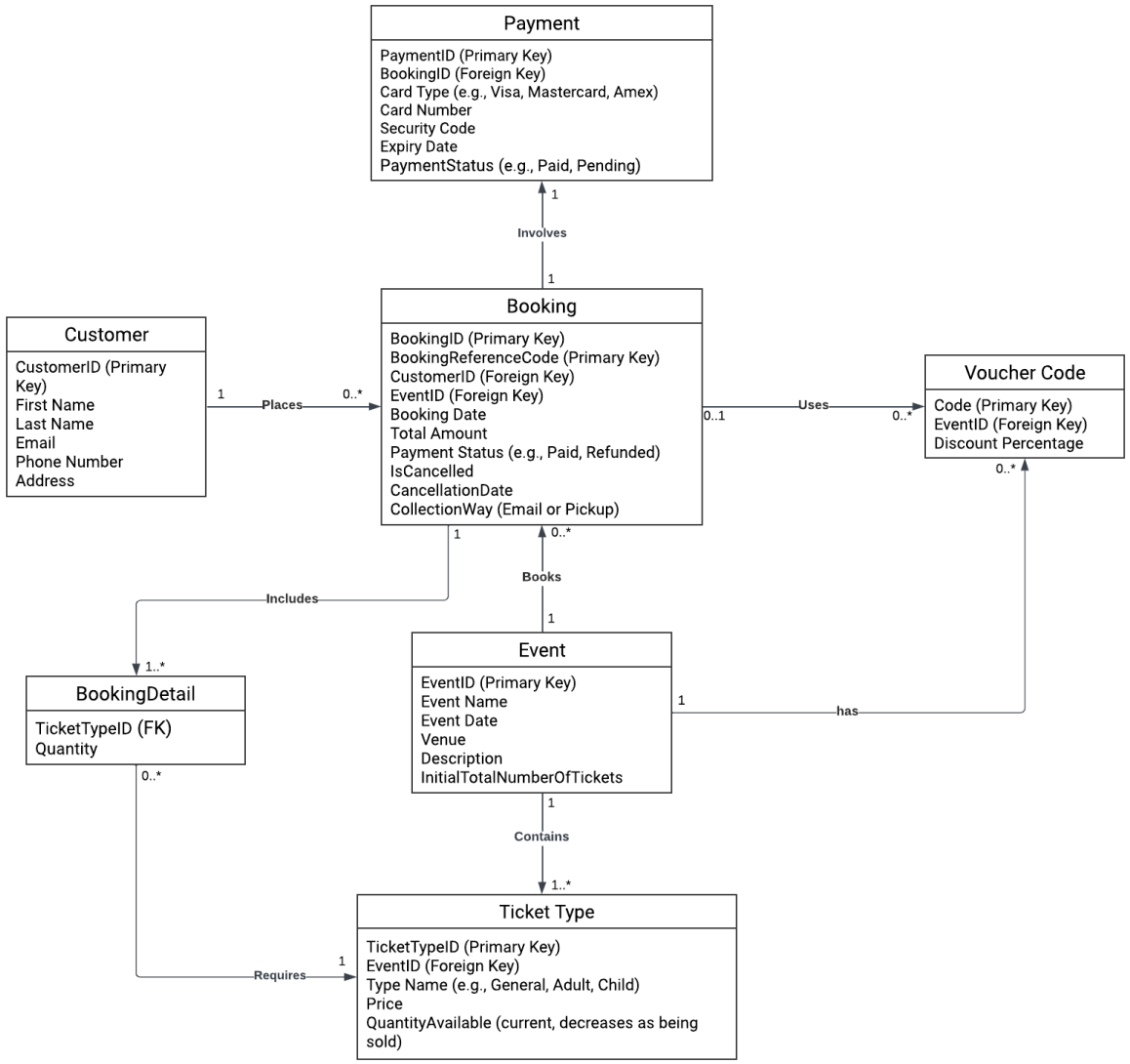
ERD



ENTITY JUSTIFICATION

Customer Entity

Starting with the Customer entity, it represents individuals who engage with the online ticket booking system. The entities Customer, Booking, and Voucher Code have a mandatory participation constraint, ensuring that a customer must be associated with at least one booking or voucher code. These entities are disjoint, signifying that a customer cannot simultaneously be part of a booking and a voucher code. This specialization is justified by the distinct functionalities and relationships each entity has within the system.

Attributes:

- customerID: Unique identifier for each customer.
- firstName, lastName: Customer's name for personalization.
- email: Contact information for communication.

- phone: Additional contact information.

Booking Entity

Moving to the Booking entity, it represents the core aspect of the system, where customers make reservations for events. The relationship between Customer and Booking is labeled as "Places", signifying that a customer places a booking. This relationship is one-to-many, reflecting that one customer can place multiple bookings. The Booking entity includes essential attributes such as bookingID and totalAmount for unique identification and tracking the booking's cost.

Attributes:

- bookingID: Unique identifier for each booking.
- customerID: Unique identifier for the customer that made the booking.
- bookingReferenceCode: Unique code for referencing bookings.
- bookingDate: Date when the booking is made.
- status: Current status of the booking (e.g., pending, confirmed).
- totalAmount: Total cost of the booking.
- isCancelled: Specifies if the booking was cancelled.
- CancellationDate: The date when the booking was cancelled.
- CollectionWay: Collection way (e.g., email, pickup).
- PaymentStatus: Current payment status of the booking (e.g. paid, refunded).

Voucher Code Entity

The Voucher Code entity represents the discount codes that customers can apply to their bookings. The relationship between Booking and Voucher Code is labeled as "Uses", indicating that a booking uses a voucher code. This relationship is many-to-one, meaning many bookings can use the same voucher code for a discount.

Attributes:

- voucherCodeID: Unique identifier for each voucher code.
- code: Alphanumeric code for applying discounts.
- discountPercentage: Percentage of the discount applied.

Event Entity

The Event entity captures details about the events for which customers can book tickets. The relationship between Customer and Event is labeled as "Books", reflecting that customers book tickets for events. This relationship is many-to-many, allowing many customers to book tickets for different events, and an event can be booked by many customers.

Attributes:

- eventID: Unique identifier for each event.
- eventName: Name of the event (e.g., circus, concert).
- eventDate: Date when the event takes place.
- venue: Location where the event is held.
- eventType: Categorizes the type of event.
- InitialTotalNumberOfTickets: Specifies the initial total number of tickets (all types together)

Ticket Type Entity

The Ticket Type entity represents the various types of tickets available for events. The relationship between Event and Ticket Type is labeled as "Contains", indicating that an event contains multiple ticket types. This relationship is one-to-many, as an event may have different ticket types with varying prices.

Attributes:

- ticketTypeID: Unique identifier for each ticket type.
- eventID: Unique identifier for the event.
- typeName: Descriptive name for the ticket type.
- price: Cost of each ticket type.
- quantityAvailable: Number of available tickets for the type.

Booking Detail Entity

The Booking Detail entity captures specific details about each booking, such as the quantity of tickets for a particular type. The relationship between Booking and Booking Detail is labeled as "Includes", indicating that a booking includes multiple booking details (ticket types). This relationship is one-to-many.

Attributes:

- bookingDetailID: Unique identifier for each booking detail.
- quantity: Number of tickets for a specific type.

Payment Entity

The Payment entity represents the financial transaction associated with a booking. The relationship between Booking and Payment is labeled as "Involves", reflecting that a booking involves a payment with the payment details stored. This relationship is one-to-one.

Attributes:

- paymentID: Unique identifier for each payment.
- cardType: Type of credit/debit card used.
- cardNumber: Card number for the transaction.
- securityCode: Security code for card verification.
- expiryDate: Expiry date of the card.
- PaymentStatus: Status of a specific payment (e.g. paid, pending).

RELATIONSHIP JUSTIFICATION

Places

- Connects Customer to Booking.
- Justification: Customers can place multiple bookings, creating a one-to-many relationship. One customer can have multiple bookings associated with their account.

Uses

- Connects Booking to Voucher Code.
- Justification: Many bookings can utilize the same voucher code for a discount. This creates a many-to-one relationship where multiple bookings are linked to a single voucher code.

Books

- Connects Customer to Event through Booking.
- Justification: Many customers can book tickets for different events. This establishes a many-to-many relationship between customers and events, mediated by the booking entity.

Includes

- Connects Booking to Booking Detail.
- Justification: One booking can include multiple booking details, such as various ticket types. This results in a one-to-many relationship where a booking comprises several booking details.

Contains

- Connects Event to Ticket Type.
- Justification: An event contains multiple ticket types with varying prices. This establishes a one-to-many relationship, as an event can have multiple ticket types associated with it.

Requires

- Connects Booking Detail to Ticket Type.
- Justification: Specifies the ticket types required in each booking. This establishes a one-to-many relationship between booking details and ticket types, as one booking detail can specify multiple required ticket types.

Involves

- Connects Booking to Payment.
- Justification: A booking involves a payment with the payment details stored. This creates a one-to-one relationship between a booking and its associated payment. Each booking is linked to a single payment.

Relational Model

Customer(CustomerID, FirstName, LastName, Email, Phone Number, Address)
Primary Key: CustomerID

Event(EventID, EventName, EventDate, Venue, Description, InitialTotalNumberOfTickets)
Primary Key: EventID

TicketType(TicketTypeID, EventID, TypeName, Price, QuantityAvailable)
Primary Key: TicketTypeID
Foreign Key: EventID references Event(EventID)

Booking(BookingID, BookingReferenceCode, CustomerID, EventID, BookingDate, TotalAmount, PaymentStatus, IsCancelled, CancellationDate, CollectionWay)
Primary Key: BookingID
Foreign Key: CustomerID references Customer(CustomerID)
Foreign Key: EventID references Event(EventID)

BookingDetail(BookingDetailID, BookingID, TicketTypeID, Quantity)
Primary Key: BookingDetailID
Foreign Key: BookingID references Booking(BookingID)
Foreign Key: TicketTypeID references TicketType(TicketTypeID)

Payment(PaymentID, BookingID, CardType, CardNumber, SecurityCode, ExpiryDate, PaymentStatus)
Primary Key: PaymentID
Foreign Key: BookingID references Booking(BookingID)

VoucherCode(Code, EventID, DiscountPercentage)
Primary Key: Code
Foreign Key: EventID references Event(EventID)

Applies(BookingID, Code)
Primary Key: (BookingID, Code)
Foreign Key: BookingID references Booking(BookingID)
Foreign Key: Code references VoucherCode(Code)

The Applies table is necessary to resolve the many-to-many relationship between Booking and VoucherCode.