Database Design for Gadget Galore

Gadget Galore is a user-friendly website allows customers to buy any electronic items .

Entities :-

Let’s identify the entities of the Gadget Galore

1. **User**
2. **Customer**
3. **Product**
4. **Category**
5. **Cart**
6. **Order**
7. **Address**
8. **Payment**
9. **Feedback**

Now let’s identify the attributes and relationships of each entity for the Gadget Galore

Roles and Permissions

1. User

Common for Both Roles: Customer and Admin

2. Customer Role

Customer:

- **Register**: New users can register by providing necessary details like username, password, email, etc.
- **Login/Logout**: Users can log in and log out of the system.
- **Update Profile**: Users can update their personal information.
- **View Orders**: Users can view their past and current orders.
- **Provide Feedback**: Users can provide feedback on products they have purchased.
- **Manage Shopping Cart**: Users can add, edit, or remove items in their shopping cart.

Admin:

- **Manage Users**: Admins can view and manage all users, including customers . This includes updating user roles and details.

3. Product
Customer:

- **View Product Details**: Users can view detailed information about a product, including description, specifications, price, and stock level.
- **Search Products**: Users can search and filter products based on various criteria.

Admin:

- **Add/Edit/Remove Products**: Admins can manage the product catalog by adding, editing, or removing products.

4. Category
Customer:

- **View Categories**: Users can view available product categories for easier navigation and filtering.

Admin:

- **Add/Edit/Remove Categories**: Admins can manage product categories, including adding new categories, editing existing ones, or removing them.

5. Order
Customer:

- **Place Order**: Users can place orders for products in their shopping cart.
- **View Order Details**: Users can view the details of their orders, including the products, quantities, and prices.
- **Order Status**: Users can view the status of their orders (e.g., pending, shipped, delivered).

Admin:

- **Manage Orders**: Admins can view all orders placed by customers, update order statuses (e.g., processing, shipped, delivered), handle cancellations.

6. Cart
Customer:

- **Add Items to Cart**: Users can add products to their shopping cart.
- **View Cart**: Users can view the contents of their shopping cart, including a summary of items and total cost.
- **Update Cart Items**: Users can edit quantities or remove items from their shopping cart.
- **Apply Discounts**: Users can apply discount codes or promotions to their shopping cart.

Admin:

- **View Carts**: Admins can view the contents of customers' shopping carts for analysis purposes.

7. Address

Customer:

- **Manage Addresses**: Users can add, edit, and delete their shipping and billing addresses.

Admin:

- **View Addresses**: Admins can view customers' addresses for order fulfilment and support purposes.

8. Feedback/Review

Customer:

- **Provide Feedback**: Users can provide feedback and rate products they have purchased.
- **View Feedback**: Users can view feedback and ratings provided by other users for products.

Admin:

- **Manage Feedback**: Admins can view, respond to, and manage customer feedback (e.g., removing inappropriate comments).

9. Payment

Customer:

- **Process Payment**: Users can handle secure payment processing for their orders.
- **View Payment Details**: Users can view payment details and confirmation for their orders.
- **Multiple Payment Methods**: Users can choose from various payment methods such as credit cards, UPI, and Cash on Delivery.

Admin:

- **View Payments**: Admins can view all payment transactions for analysis or record-keeping purposes.

Tables & Relationship

1. User Table

User_ID (Primary Key)

Username

Password

Email

first_name

last_name

date_of_birth

FOREIGN KEY (role_id) REFERENCES UserRole(id)

Child: UserRole (Many-to-One)

Parent: Order, Address, Feedback/Review, Cart (One-to-Many)

2. Customer Table

CustomerID (Primary Key, Foreign Key referencing User.UserID)

UserID (Foreign Key referencing User.UserID)

Child: User (Many-to-One)

3. Product Table

ProductID (Primary Key)

Name

Description

Price

Images

catID (Foreign Key referencing Category.CategoryID)

StockQuantity

Parent: Category (Many-to-One)

Child: Feedback/Review, Cart (One-to-Many)

4. Category Table

CategoryID (Primary Key)

Name

Description

Child: Product (One-to-Many)

5. Order Table

OrderID (Primary Key)

UserID (Foreign Key referencing User.UserID)

OrderDate

TotalAmount

PaymentStatus

OrderStatus

ShippingAddress (Foreign Key referencing Address.AddressID)

BillingAddress (Foreign Key referencing Address.AddressID)

cID (Foreign Key referencing Cart.CartID)

Child: User (Many-to-One)

Child: Address (Many-to-One)

Child: Cart (Many-to-One)

Parent: Payment (One-to-Many)

6. Cart Table

CartID (Primary Key)

UserID (Foreign Key referencing User.UserID)

Products (JSON or serialized format)

Child: User (Many-to-One)

Parent: Order (One-to-Many)

7. Address Table

AddressID (Primary Key)

UserID (Foreign Key referencing User.UserID)

Landmark

Address

City

State

Pincode

Child: User (Many-to-One)

Parent: Order (One-to-Many)

8. Feedback/Review Table

FeedbackID (Primary Key)

UserID (Foreign Key referencing User.UserID)

ProductID (Foreign Key referencing Product.ProductID)

Rating

Comment

FeedbackDate

Child: User (Many-to-One)

Child: Product (Many-to-One)

9. Payment Table

PaymentID (Primary Key)

OrderID (Foreign Key referencing Order.OrderID)

PaymentDate

Amount

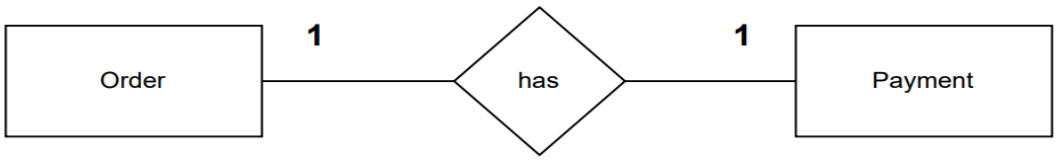
PaymentMethod

PaymentStatus

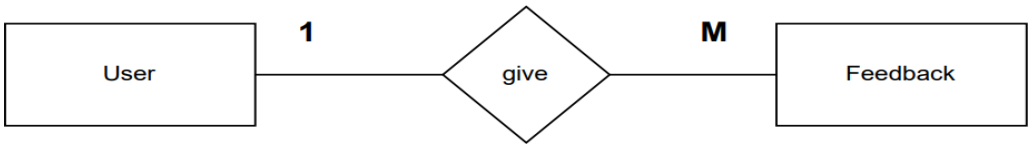
Child: Order (Many-to-One)

Let’s see a few examples of relationships:

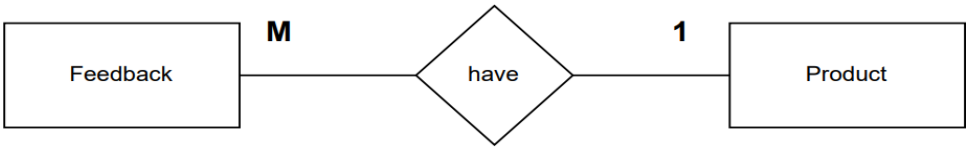
One to One
Order to Payment



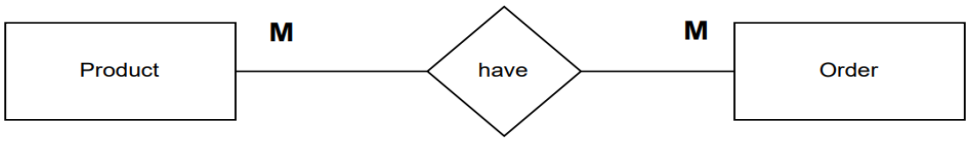
One to Many
User to Feedback



Many to One
Feedback to Product



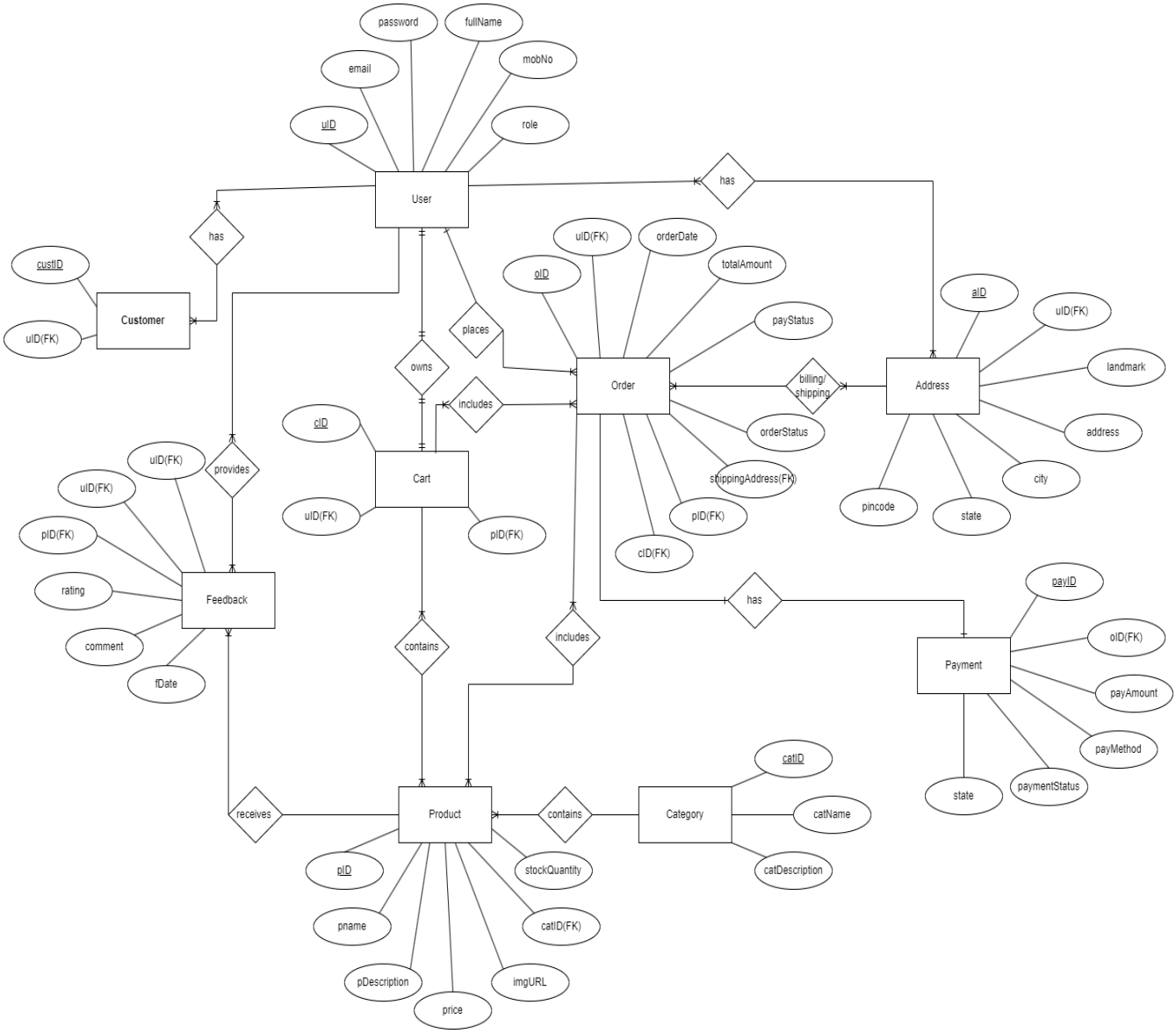
Many to Many
Product to Order



Now, let’s create the ER diagram to visually represent the entities and relationships.

ERD Diagram

In this ERD:



4. Creating a Database

Using MySQL server, create a new database for your student management system. You can do this with SQL commands or through the graphical interface.

Create database

```
CREATE DATABASE IF NOT EXISTS Gadget Galore;
```

Use the database

```
USE Gadget Galore;
```

```
CREATE TABLE User (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    FullName VARCHAR(255) NOT NULL,
    PhoneNumber VARCHAR(15),
    Role ENUM('customer', 'admin') NOT NULL
);

CREATE TABLE Customer (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE
);

CREATE TABLE Category (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Description TEXT
);

CREATE TABLE Product (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Description TEXT,
    Price DECIMAL(10, 2) NOT NULL,
    Images JSON,
    CategoryID INT,
    StockQuantity INT NOT NULL,
    FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID) ON DELETE SET
NULL
);

CREATE TABLE Address (
    AddressID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    UserID INT,
    Landmark VARCHAR(255),
    Address TEXT NOT NULL,
    City VARCHAR(100) NOT NULL,
    State VARCHAR(100) NOT NULL,
    Pincode VARCHAR(10) NOT NULL,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE
);

CREATE TABLE `Order` (
    OrderID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT,
    OrderDate DATETIME NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    PaymentStatus ENUM('pending', 'completed', 'failed') NOT NULL,
    OrderStatus ENUM('processing', 'shipped', 'delivered') NOT NULL,
    ShippingAddressID INT,
    BillingAddressID INT,
    CartID INT,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,
    FOREIGN KEY (ShippingAddressID) REFERENCES Address(AddressID) ON DELETE
SET NULL,
    FOREIGN KEY (BillingAddressID) REFERENCES Address(AddressID) ON DELETE
SET NULL,
    FOREIGN KEY (CartID) REFERENCES Cart(CartID) ON DELETE SET NULL
);

CREATE TABLE Cart (
    CartID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT,
    Products JSON,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE
);

CREATE TABLE Feedback (
    FeedbackID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT,
    ProductID INT,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    Comment TEXT,
    FeedbackDate DATETIME NOT NULL,
    FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE,
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID) ON DELETE
CASCADE
);
```

```
CREATE TABLE Payment (
    PaymentID INT AUTO_INCREMENT PRIMARY KEY,
    OrderID INT,
    PaymentDate DATETIME NOT NULL,
    Amount DECIMAL(10, 2) NOT NULL,
    PaymentMethod ENUM('Credit Card', 'UPI', 'Cash on Delivery') NOT NULL,
    PaymentStatus ENUM('completed', 'pending', 'failed') NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID) ON DELETE CASCADE
);
```