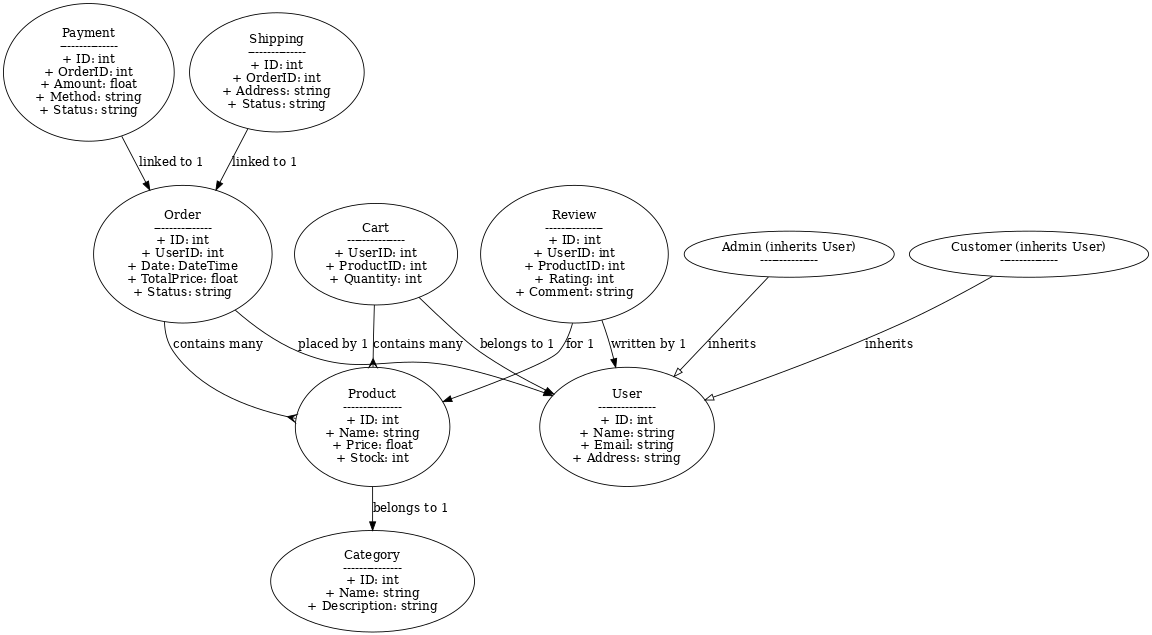
### **Project Requirement 2 - Software Architecture Document (SAD)**

**Software Architecture Document**

1. **Product Overview:**
   * **Vision:** Build a scalable and user-friendly e-commerce platform for online shopping.
   * **Stakeholders:** Business owners, developers, and end-users (customers).
   * **Target Market:** Small-to-medium-sized businesses selling physical or digital products.

2.

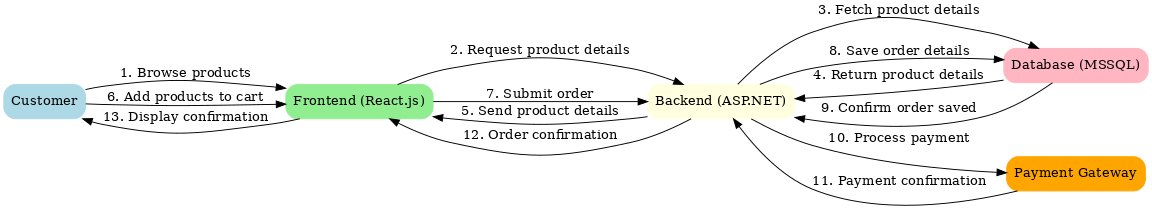
**Static :**



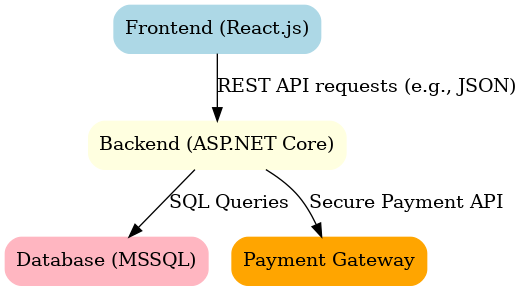
**Dynamic :**

**Example Sequence Diagram for Placing an Order:**

1. Customer browses products (Frontend sends request to Backend).
2. Backend retrieves product data from the database and returns it to the frontend.
3. Customer adds items to the cart.
4. Frontend sends a "Create Order" request to the Backend.
5. Backend processes the order and updates the database.
6. Payment information is sent to the Payment Gateway.
7. The payment is processed, and confirmation is returned.



**System Architecture :**

****

**Mapping between models**

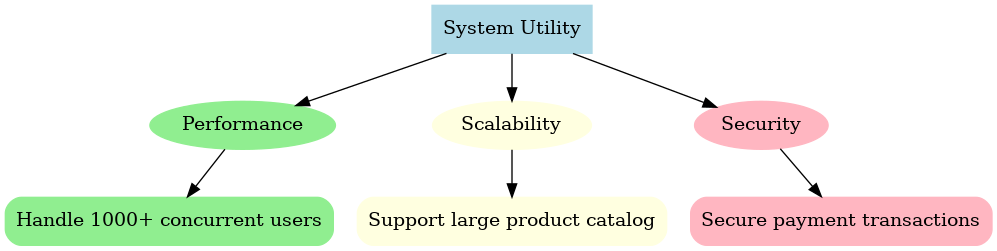
| **Class** | **Dynamic Interaction** |
| --- | --- |
| **Product** | **Fetch product details** |
| **Cart** | **Add/remove products** |
| **Order** | **Place an order** |
| **Payment** | **Validate payment and process order** |

**Architectural Design Rationale:**

* Layered Architecture:
  + Presentation Layer: Handles user interactions (React.js frontend).
  + Business Logic Layer: Processes requests and applies logic (ASP.NET Core backend).
  + Data Access Layer: Manages database operations (MSSQL).
* Justification:
  + Scalability: Layers allow independent scaling of the frontend and backend.
  + Maintainability: Separation of concerns ensures easier debugging and updates.
  + Security: Sensitive operations (e.g., payments) are isolated in the backend.

**Utility Tree:**

* Scenarios:
  + Performance: The system should handle 1000 concurrent users during peak sales.
  + Scalability: Add new product categories without impacting existing functionality.
  + Security: Protect user payment information with encryption.
* Detailed Scenario:
  + Performance during Black Friday sales:
    - Utility: High performance.
    - Quality Attribute: Fast response under heavy traffic.
    - Test Case: Simulate 1000 concurrent requests to the product listing API.

****