**Scoop architecture**

 **Frontend:** This is the user interface where customers interact with the system. It can be a website, mobile app, or kiosk at the cinema. The frontend displays movie listings, showtimes, seat selection options, and allows users to purchase tickets securely.

 **API Gateway:** This acts as a single entry point for all requests coming from the frontend. It routes these requests to the appropriate backend services.

 **Ticketing Service:** This core service manages ticket inventory, seat selection, and pricing. It keeps track of available seats for each showtime and handles the booking process.

 **User Management Service:** This service handles user accounts, logins, and profiles. It stores user information securely and allows for features like purchase history and saved preferences.

 **Payment Gateway:** This secure service processes customer payments. It connects the system to a payment processor like Stripe or PayPal to handle transactions securely.

 **Showtime & Movie Data:** This could be a separate service or integrated with the ticketing service. It stores information about movies, showtimes, trailers, and descriptions.

 **Scalability:** The system can be easily scaled to handle increased traffic by adding more servers to each service layer.

 **Security:** User data and financial information are kept secure by using separate services for authentication and payments.

 **Maintainability:** Each service is independent and can be updated or maintained without affecting other parts of the system.

**Data Storage:**

* **Database:** A robust database like MySQL or PostgreSQL is typically used to store movie information, showtimes, seat layouts, user accounts, booking details, and transaction history.
* **NoSQL Databases:** For real-time features like seat selection updates, NoSQL databases like Redis can be used to store and manage seat availability efficiently.

**Communication Protocols:**

* **RESTful APIs:** These APIs define a standard way for the frontend and backend services to communicate. They use HTTP verbs like GET, POST, PUT, and DELETE to request and manipulate data.
* **Messaging Queues:** For asynchronous communication, message queues like RabbitMQ can be used. This allows services to send and receive messages without waiting for a direct response, improving overall system performance.

**Security Considerations:**

* **Authentication and Authorization:** User accounts should be secured with strong password hashing and multi-factor authentication. User roles and permissions should be implemented to control access to different functionalities.
* **Payment Security:** The payment gateway should be PCI DSS compliant to ensure secure handling of credit card information. Data encryption should be used throughout the system to protect sensitive data.