**ARCHITECTURE DOCUMENT**

The IndiaRide system follows a Microservices Architecture with a Django backend and HTML, CSS, JavaScript frontend. It incorporates event-driven communication, serverless computing, and RESTful APIs for seamless operations. Data is stored in PostgreSQL/MySQL and MongoDB. Deployed on cloud infrastructure using Docker and Kubernetes, it ensures scalability, security, high availability, and optimal performance.

1. **Application**
   1. **Microservices**

The IndiaRide platform is designed using a Microservices Architecture to ensure modularity, scalability, and efficient management of services. The Django backend handles authentication, vehicle management, booking processing, and payments, while the frontend is built using HTML, CSS, and JavaScript, ensuring a responsive user experience.

**Justification for Microservices Architecture:**

* Enables independent development and deployment of different services.
* Improves system resilience and fault isolation.
* Facilitates scalability, allowing specific services to scale based on demand.
* Supports diverse technology stacks for different services.

1. **Event-Driven Mechanisms**

IndiaRide utilizes an Event-Driven Architecture for real-time data processing and notification handling.

* Events: Booking confirmations, payment status updates, vehicle availability changes.
* Message Broker: Uses Kafka or RabbitMQ to ensure asynchronous and reliable communication.

1. **Serverless Computing**

Certain non-critical, event-driven processes such as notification dispatch (email/SMS alerts) and log analysis are implemented using Serverless Functions (e.g., AWS Lambda, Azure Functions) to optimize resource utilization.

1. **Database Design**

* Database Type: Relational Database (PostgreSQL/MySQL) for structured data and MongoDB for unstructured vehicle metadata.
* Entity-Relationship (ER) Diagram: Illustrates entities such as Users, Vehicles, Bookings, Payments, and Ratings.
* Schema Design: Includes primary keys, foreign keys, and relationships between entities.

1. **Data Exchange Contract**

* Frequency of Data Exchange: Real-time and scheduled batch updates for analytics.
* Data Sets: User details, vehicle listings, booking transactions, and payment records.
* Mode of Exchange: RESTful APIs for synchronous communication and message queues for event-driven actions.

1. **Architecture Diagrams**

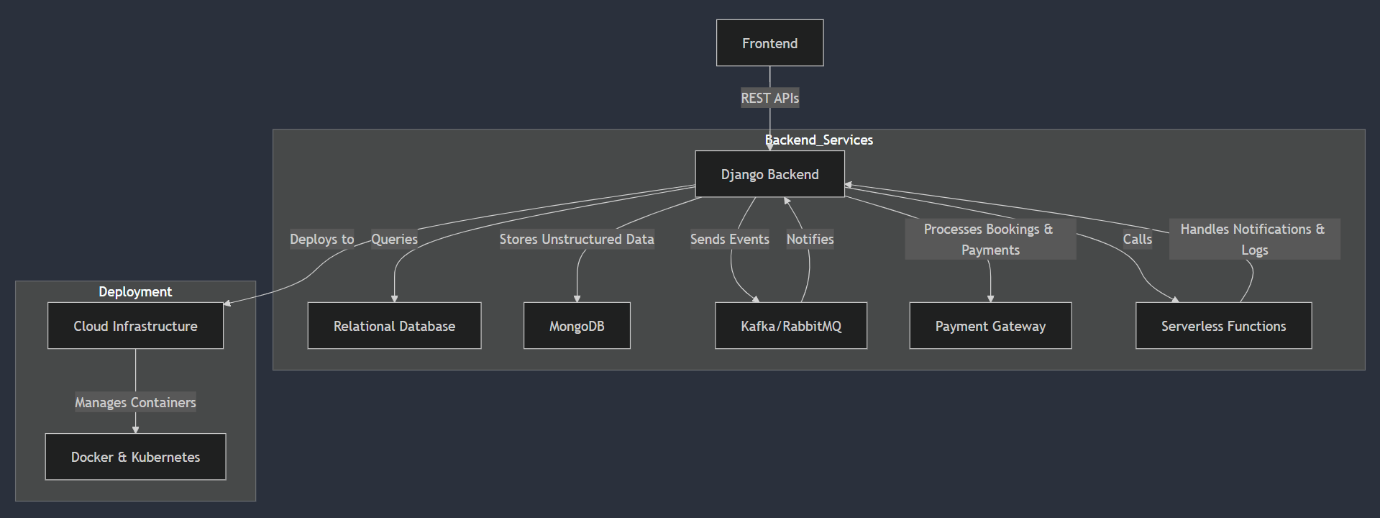
To visualize and validate the system design, the following diagrams are included:

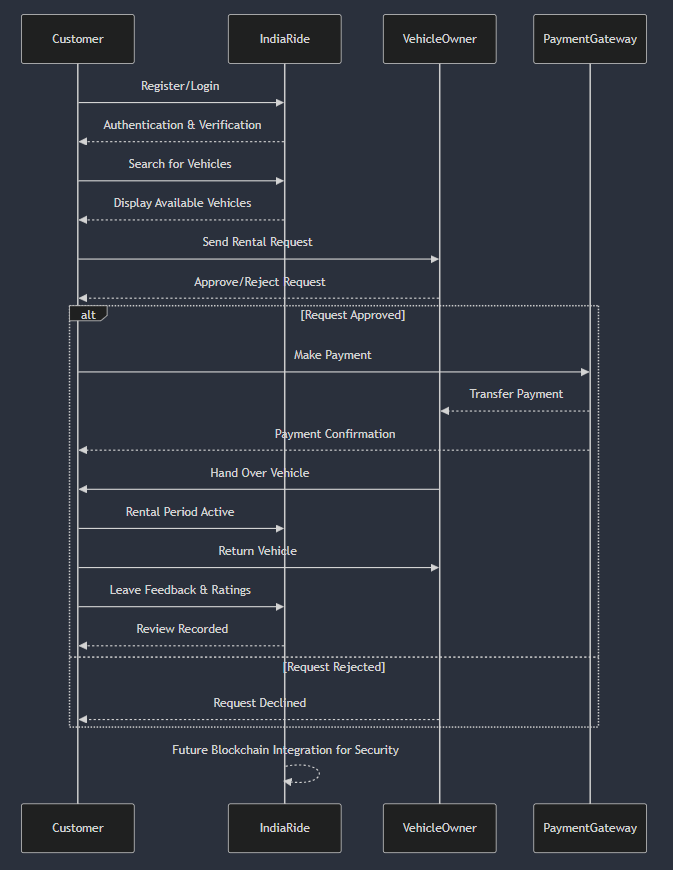
* Class Diagram: Defines system entities and their relationships.
* Sequence Diagram: Details step-by-step interactions during booking and payment processes.
* State Diagram: Represents the different states a booking goes through (e.g., Initiated → Payment Pending → Confirmed → Completed).
* Flow Diagram: Shows the step-by-step flow of how a user interacts with the system, from browsing to booking and payment.

7. **Deployment Strategy**

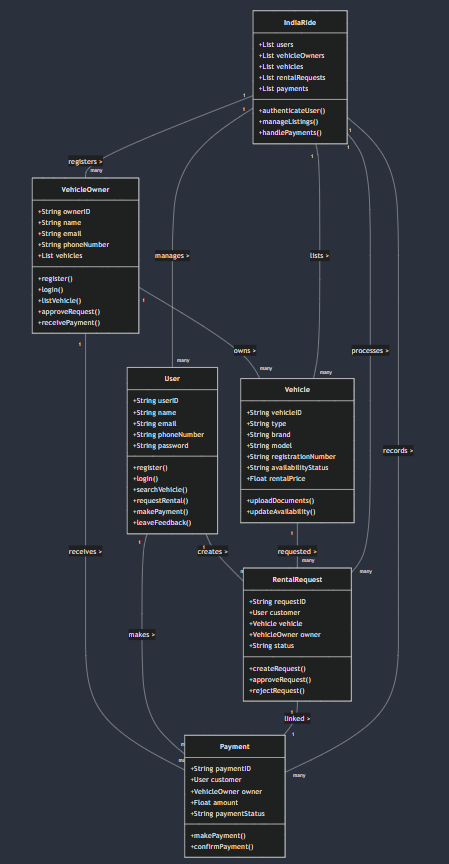
The system is deployed on cloud infrastructure (AWS/Azure/GCP) using Docker and Kubernetes for containerization and scalability. This ensures high availability, security, and fault tolerance, making the platform robust and adaptable to varying loads.

**DIAGRAMS**

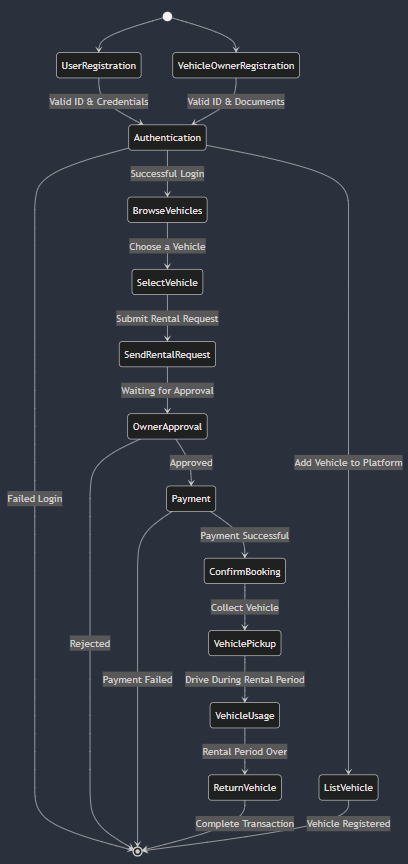
1. Architecture Diagram
2. Sequence Diagram



1. Class Diagram



1. State Diagram



1. Flow Diagram

A screenshot of a diagram

AI-generated content may be incorrect.

**Conclusion**

The IndiaRide vehicle rental system is designed to be a secure, scalable, and user-friendly platform. By leveraging Microservices, Event-Driven Architecture, and Serverless Computing, the system ensures seamless operations while offering flexibility for future enhancements. The use of Django, HTML, CSS, JavaScript, PostgreSQL/MySQL, MongoDB, and cloud technologies guarantees a reliable and high-performance platform for users.