Vehicle Rental Service – Project Documentation

# **1. Introduction**

The Vehicle Rental Service project is designed to streamline the management of vehicle rentals by providing an efficient, scalable, and easy-to-use API-based backend system. The application is built using FastAPI with MongoDB as the database, ensuring high performance and flexibility. This documentation describes the objectives, tools used, system design, and API details.

Objectives:

- Develop a RESTful API for managing vehicles and bookings.

- Store and manage data efficiently using MongoDB.

- Ensure vehicle availability is tracked accurately.

- Provide scalable architecture for future enhancements.

- Maintain clean JSON-only responses for integrations.

# **2. Tools and Technologies**

The following tools and technologies were used in the development:

- Python 3.10+: Programming language

- FastAPI: Modern, high-performance API framework

- MongoDB: NoSQL database

- Motor: Async MongoDB driver

- Uvicorn: ASGI server

- Pydantic: Data validation

- Logging: Error and event tracking

- Postman: API testing

- Anaconda / Virtual Environment: Package and environment management

**3. System Architecture**

The system follows a layered architecture with clear separation of concerns:  
 1.API Layer: Handles HTTP requests and responses.  
 2.Service Layer: Contains business logic.  
 3.Database Layer: Interacts with MongoDB using Motor.  
 4.Utilities: Logging and exception handling.  
   
The architecture ensures modularity, scalability, and maintainability.

[System Architecture Diagram Placeholder]

# **4. Database Design**

MongoDB collections used in the system:  
   
1. Vehicles Collection:  
 - id (ObjectId): Unique identifier  
 - make (string): Vehicle manufacturer  
 - model (string): Vehicle model  
 - year (int): Year of manufacture  
 - plate\_number (string): Vehicle plate number  
 - daily\_rate (float): Rental price per day  
 - is\_available (boolean): Availability status  
   
2. Bookings Collection:  
 - id (ObjectId): Unique identifier  
 - vehicle\_id (ObjectId): Linked vehicle  
 - customer\_name (string): Name of customer  
 - start\_date (date): Booking start date  
 - end\_date (date): Booking end date  
 - total\_amount (float): Total rental cost

# **5. API Endpoints**

Vehicles Endpoints:-

POST /vehicles/ - Add new vehicle  
 GET /vehicles/ - List vehicles  
 GET /vehicles/{id} - Get vehicle by ID  
 PUT /vehicles/{id} - Update vehicle  
 DELETE /vehicles/{id} - Delete vehicle

Bookings Endpoints

POST /bookings/ - Create booking  
 GET /bookings/ - List bookings

# **6. Example API Requests**

Example - Add Vehicle:

POST /vehicles/  
 {  
 "make": "Honda",  
 "model": "Civic",  
 "year": 2021,  
 "plate\_number": "XYZ987",  
 "daily\_rate": 60.0  
 }

Example - Create Booking:

POST /bookings/  
 {  
 "vehicle\_id": "66e23f7a6a2e3b5c9d123456",  
 "customer\_name": "Bob",  
 "start\_date": "2025-09-15",  
 "end\_date": "2025-09-18"  
 }

# **7. Error Handling**

The system uses FastAPI's exception handlers with custom logging. Common error codes:  
 - 422: Validation error (wrong JSON format or missing fields)  
 - 404: Resource not found (vehicle or booking not available)  
 - 500: Internal server error (unexpected issue)

# **8. Testing Strategy**

Testing was performed using Postman and FastAPI's interactive docs. Key test cases include:  
 - Adding vehicles with valid and invalid data  
 - Booking vehicles with overlapping dates  
 - Updating and deleting vehicles  
 - Validating MongoDB ObjectId values

# **9. Deployment**

Steps for deployment:  
 1. Set up a virtual environment and install dependencies.  
 2. Configure MongoDB connection string in environment variables.  
 3. Use Uvicorn/Gunicorn for serving FastAPI.  
 4. Optionally use Docker for containerization.  
 5. Deploy on cloud (AWS, Azure, or Heroku).

# **10. Security Considerations**

Security measures to be considered:  
 - JWT authentication for users  
 - Input validation using Pydantic  
 - Secure MongoDB with username/password  
 - HTTPS for all API communications  
 - Role-based access for admins and customers

# **11. Future Enhancements**

- Add JWT authentication and role-based access control.

- Integrate payment gateways for online booking.

- Provide analytics dashboard for admins.

- Add automated email/SMS notifications.

- Develop a front-end (React/Angular) for user interaction.

# **12. Conclusion**

The Vehicle Rental Service provides a reliable backend system for managing vehicles and bookings. By leveraging FastAPI and MongoDB, the system achieves high performance and scalability. Future enhancements like authentication, payments, and analytics can make it production-ready.