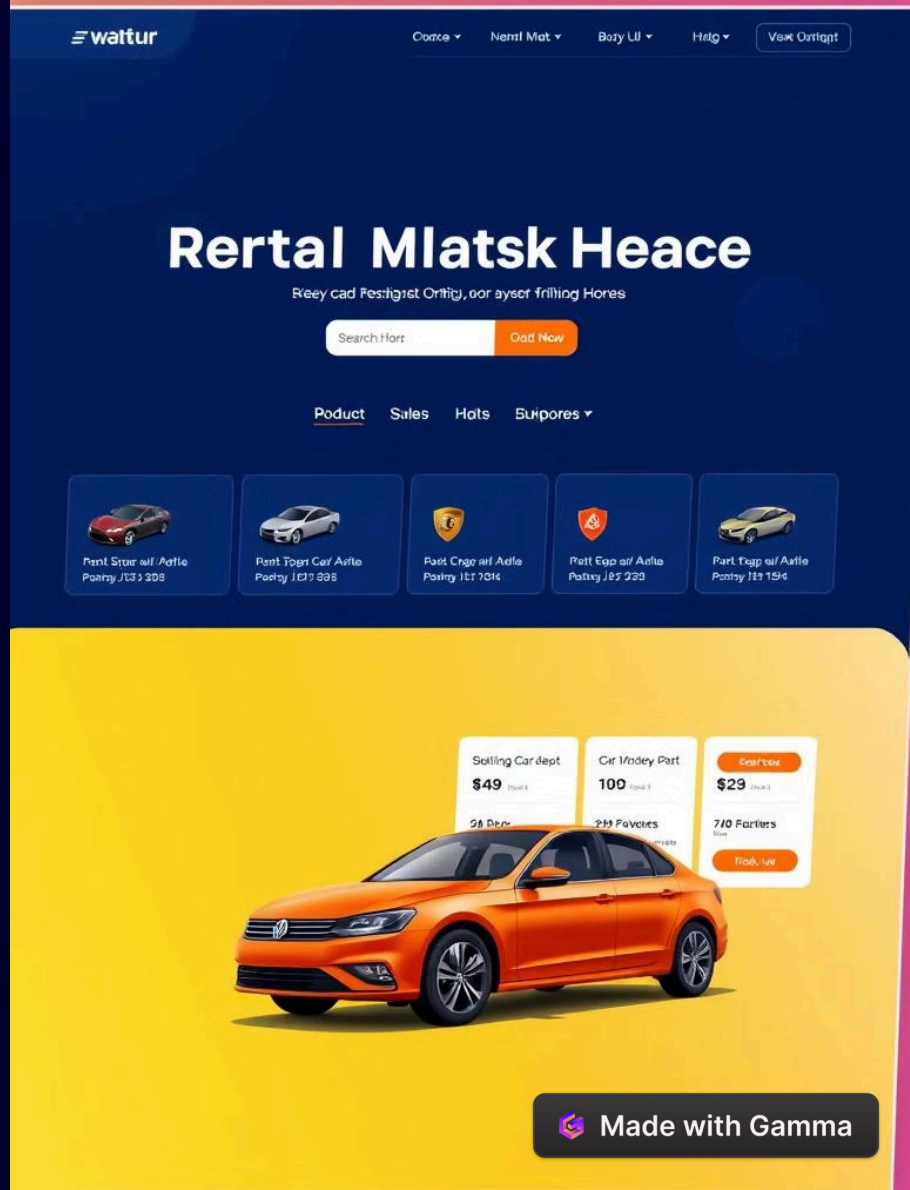
Marketplace Technical Foundation: Rental Car E-commerce

This presentation outlines the technical foundation for a user-friendly online platform that allows users to search for, book, and pay for car rentals. The system will cater to both customers and administrators, ensuring efficient management of rentals, user data, and reviews.

by **AFAQ UL ISLAM**



Step 1: Technical Requirements Definition

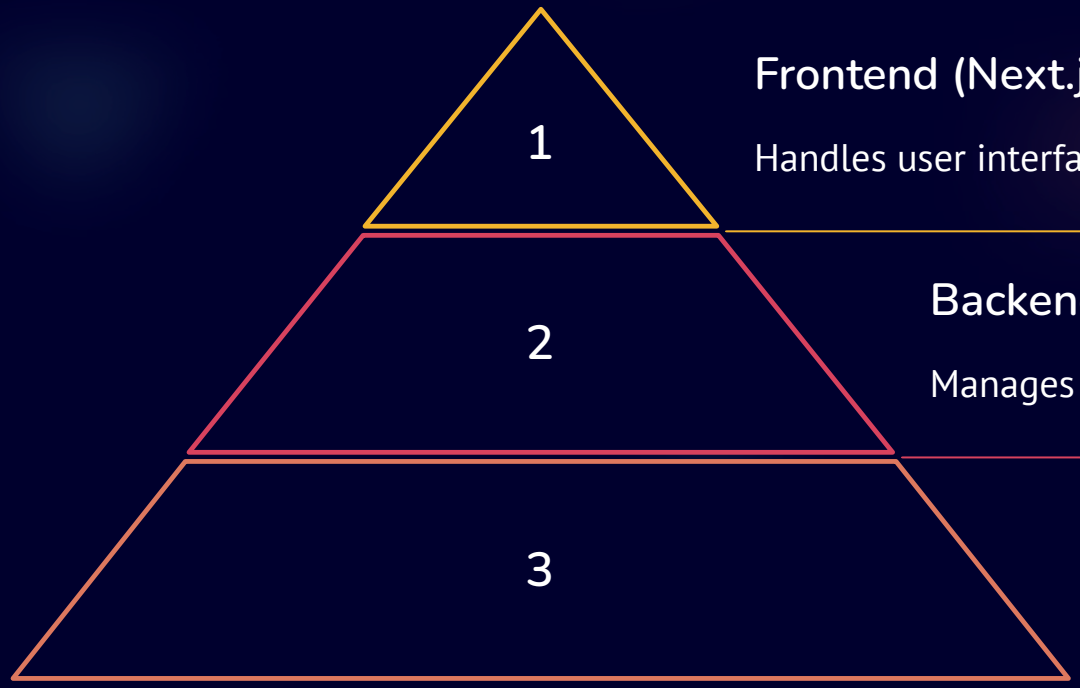
Frontend Requirements

The frontend will be built using Next.js for server-side rendering and optimized performance. The website will be mobile-friendly using CSS frameworks like Tailwind CSS or Bootstrap. User-friendly interfaces will be implemented with intuitive navigation, clear CTAs, and modals for login/signup.

Backend Requirements

Sanity CMS will be utilized for managing car listings, user data, and reviews. Third-party integrations will be implemented for payment processing through APIs like Stripe or PayPal and shipment tracking (if applicable) through relevant APIs.

Step 2: System Architecture Design



Frontend (Next.js)

Handles user interface, requests, and data display.

Backend (Sanity CMS)

Manages data storage and retrieval.

APIs

Handles payment processing and third-party services.

Example Workflows

1

User Registration

User submits registration form → Data sent to backend → Confirmation email sent.

2

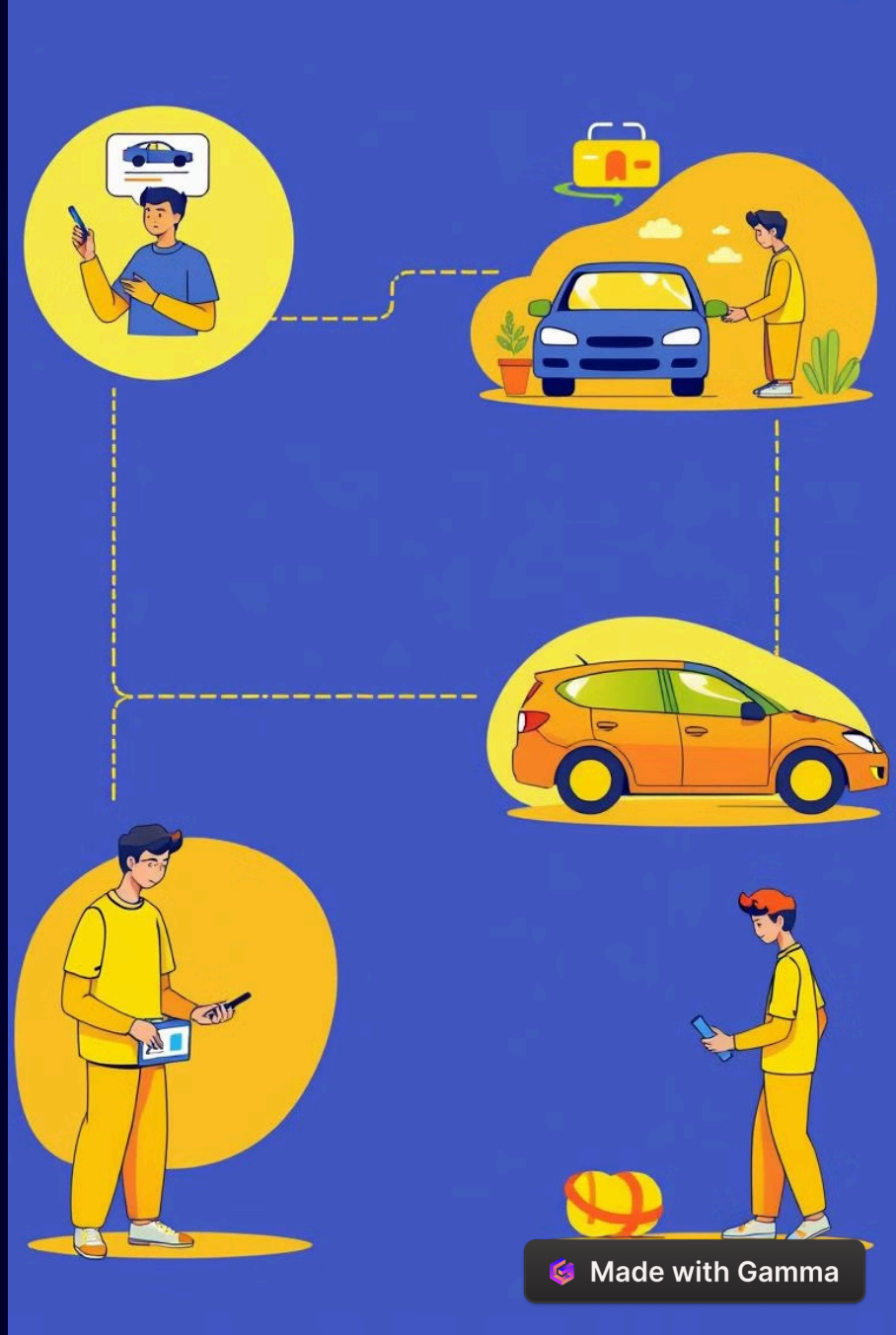
Product Browsing

User searches/filter cars → API fetches data from Sanity → Results displayed.

3

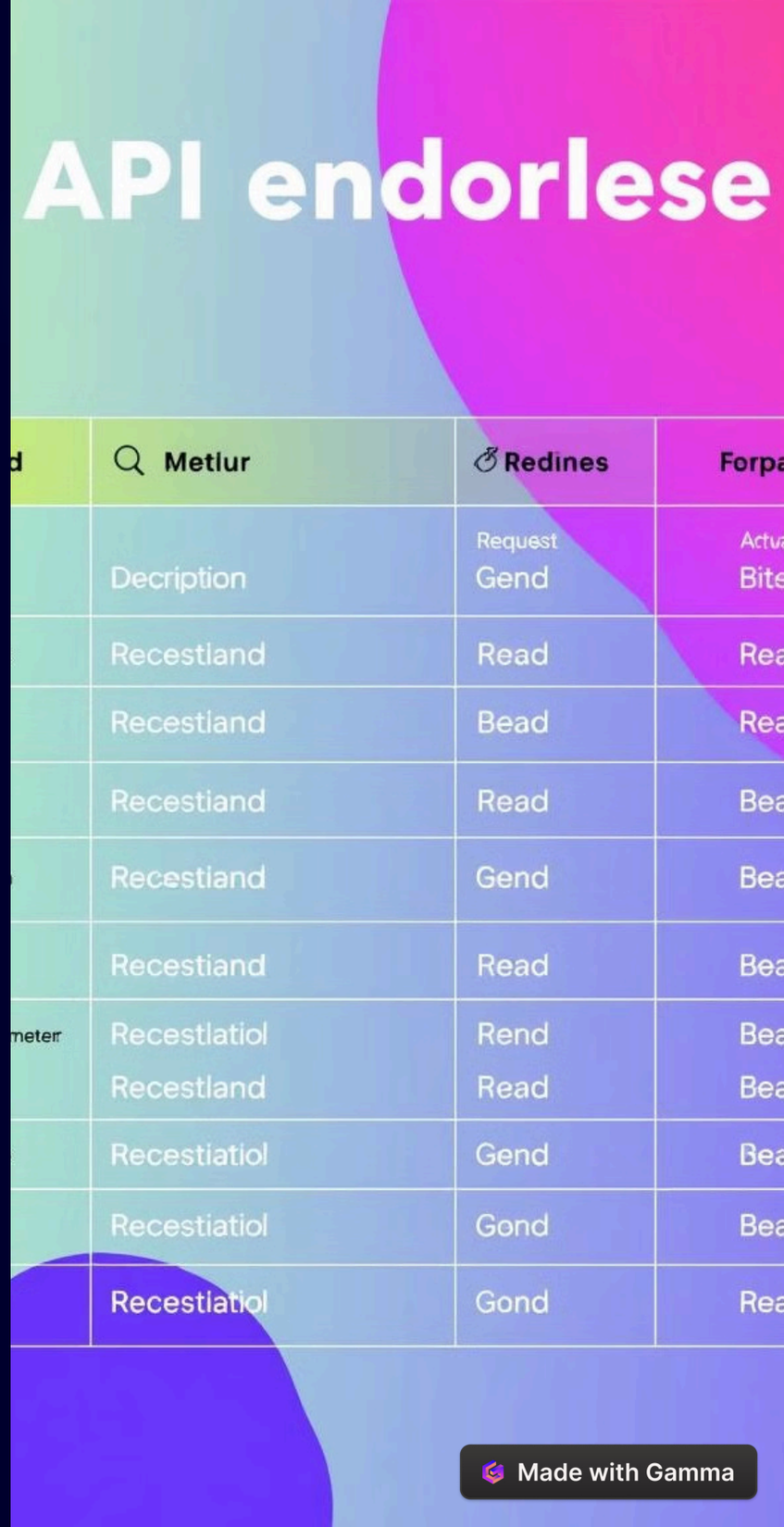
Order Placement

User selects a car → Proceeds to payment → Payment processed via API → Confirmation displayed.



Step 3: API Requirements Planning

Endpoint	Method	Description
GET /api/cars	GET	Retrieve a list of available cars with filters (type, price).
GET /api/cars/:id	GET	Retrieve detailed information about a specific car.
POST /api/rentals	POST	Create a new rental order with user details and car selection.
POST /api/payments	POST	Process payment information securely.
GET /api/reviews/:carId	GET	Fetch reviews for a specific car.



Step 4: Technical Documentation Writing

System Architecture Document

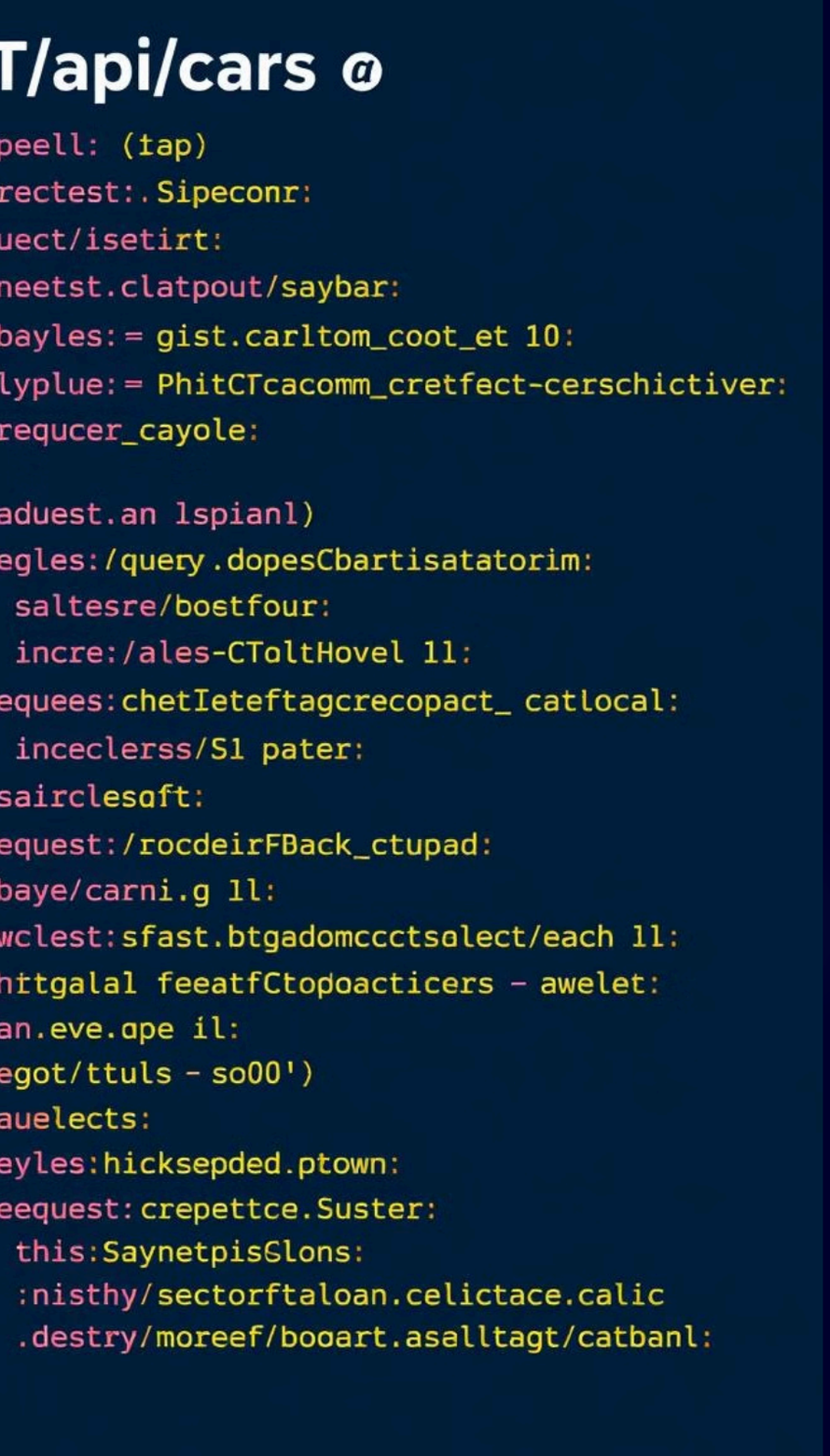
Includes diagrams showing interactions between components. Describes the purpose of each component in the architecture.

API Specifications Document

Details each endpoint, including methods, payloads, response formats, and examples.

Data Schemas Document

Defines schemas for entities such as cars, users, rentals, and reviews in Sanity CMS.



Example API Documentation Snippet

GET /api/cars

Description

Fetches a list of available cars based on filters.

Query Parameters

- type: (optional) Type of car (e.g., SUV, sedan).
- price: (optional) Price range for filtering.

Response

{ "cars": \[{ "id": "1", "model": "Toyota Camry", "price_per_day": 50, "image": "url_to_image", "features": \["Automatic", "5 seats"\] }, ... \] }

Step 5: Data Schemas Document

Schema	Field	Data Type	Constraints	Description
Car	id	string	Unique	Unique identifier for the car.
Car	make	string	Required	Manufacturer of the car (e.g., Toyota).
Car	model	string	Required	Model of the car (e.g., Camry).
Car	year	number	Required	Year of manufacture.
Car	price_per_day	number	Required	Rental price per day.
Car	image	string	Required	URL to the car's image.
Car	features	array of strings	Optional	Array of features (e.g., ["Automatic", "GPS"]).
Car	availability	object	Required	Availability status.
Car	location	object	Required	Location details for pickup.
Car	createdAt	date	Required	Timestamp of when the car was added.
Car	updatedAt	date	Required	Timestamp of last update.
User	id	string	Unique	Unique identifier for the user.
User	firstName	string	Required	User's first name.
User	lastName	string	Required	User's last name.
User	email	object	Required	User's email address.
User	passwordHash	string	Required	Hashed password for authentication.
User	phoneNumber	object	Optional	User's phone number information.
User	createdAt	date	Required	Timestamp of account creation.
User	updatedAt	date	Required	Timestamp of last update.
Rental	id	string	Unique	Unique identifier for the rental transaction.
Rental	userId	string	Required	Reference to the user's ID who rented the car.
Rental	carId	string	Required	Reference to the rented car's ID.
Rental	rentalStartDate	date	Required	Start date of the rental period.
Rental	rentalEndDate	date	Required	End date of the rental period.
Rental	totalPrice	number	Required	Total cost for the rental period.
Rental	paymentStatus	object	Required	Payment status details.
Rental	createdAt	date	Required	Timestamp of when rental was created.
Rental	updatedAt	date	Required	Timestamp of last update.
Review	id	string	Unique	Unique identifier for the review.
Review	carId	string	Required	Reference to the reviewed car's ID.
Review	userId	string	Required	Reference to the user's ID who wrote the review.
Review	rating	number	Required	Rating given by user (1-5 scale).
Review	comment	string	Optional	Review text provided by user.
Review	createdAt	date	Required	Timestamp of when review was created.
Review	updatedAt	date	Required	Timestamp of last update.

Business Model Considerations

Consider implementing a commission-based revenue model where fees are charged per rental. Additional fees may include delivery charges or subscription packages for frequent users. This approach can enhance revenue stability while providing value to customers.