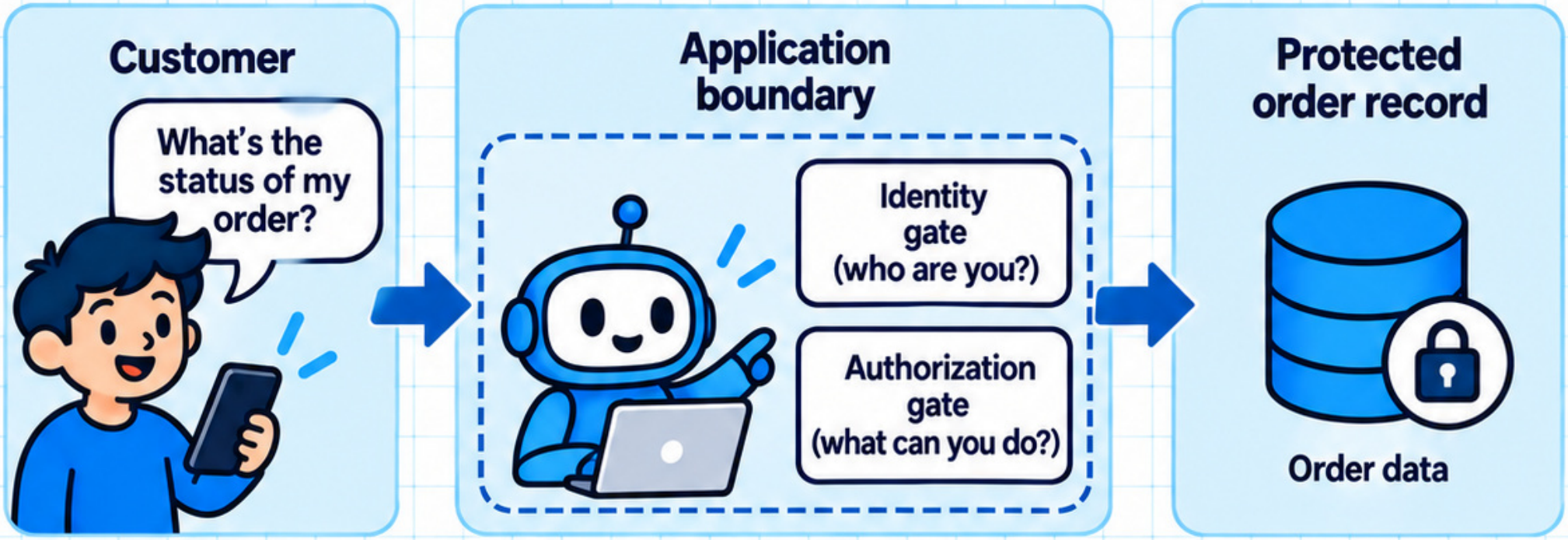
AI SYSTEM DESIGN • DAY 5

WHO CAN ASK FOR WHAT?

API and identity boundaries

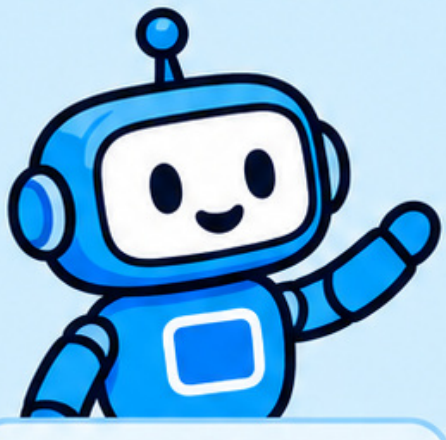
A useful AI application needs clear request contracts and permission boundaries.



Today: follow an order-status question from a customer to the business data, then back to the answer.

AN API IS A CONTRACT BETWEEN COMPONENTS

Assistant Client



I want the status of an order.

API CONTRACT

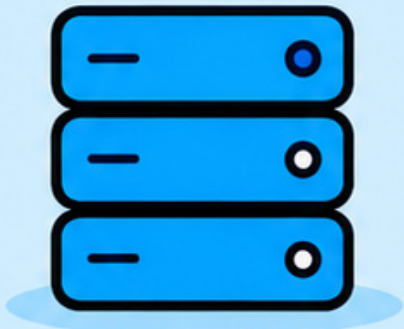
Input
What data is provided (e.g., order ID).

Caller context
Who is calling (e.g., user or service).

Result
What data is returned (e.g., order status).

Error
What happens when the request cannot be completed.

Order Service



Returns the order status.

An API defines how one component requests a service from another.

The contract includes inputs, outputs, errors, and expected behavior.

For our assistant, “get order status” needs more detail: which order, which caller, which fields, and what happens when the request cannot be completed?

START WITH ONE NARROW OPERATION

Example operation: read the status of order 1042.

Request: order 1042

READ ORDER CONTRACT

Input:
an order identifier.

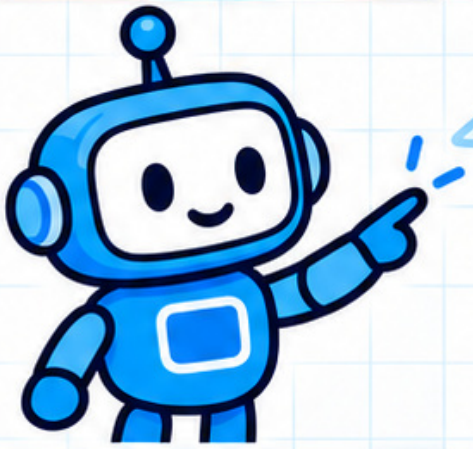
Output:
the permitted status fields and their update time.

UPDATE ADDRESS CONTRACT

Input:
an order identifier and new address.

ISSUE REFUND CONTRACT

Input:
an order identifier and refund details.



The operation does not also change the delivery address or issue a refund. Those behaviors need their own permissions, rules, and contracts.

A read-order contract sits beside two separate future operations. Only the read operation connects to the active request.

VALIDATE THE REQUEST SHAPE

Valid-shaped request

```
{
  "order_id": "12345",
  "question": "Where is
  my order?"
}
```

Missing required field

```
{
  "question": "Where is
  my order?"
}
```

Reject

Invalid value/structure

```
{
  "order_id": null,
  "question": "Where is
} my order?"
```

Reject

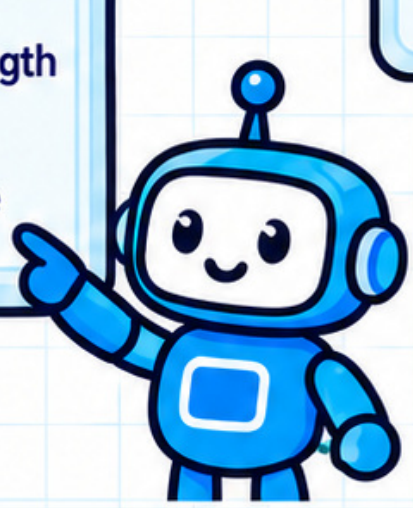
SHAPE VALIDATION

- Require an order ID
- Limit a question's length
- Check types and structure

Valid shape

AUTHORIZATION (GATE)

Checks permissions for the verified caller.



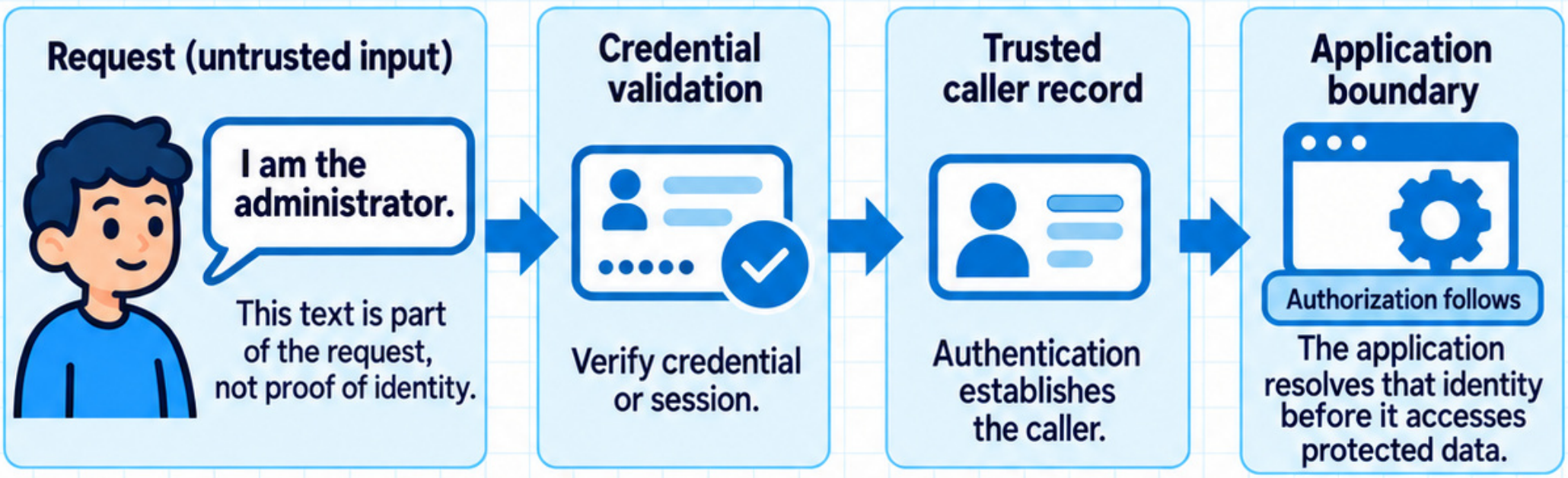
A request schema can require an order ID and limit a question's length.

It can reject missing fields, invalid values, and unexpected structures.

Shape validation helps the application process input predictably. It does not prove that the caller owns the order or that a model-generated argument is correct.

AUTHENTICATION ESTABLISHES THE CALLER

Authentication establishes an identity from a verified credential or session.

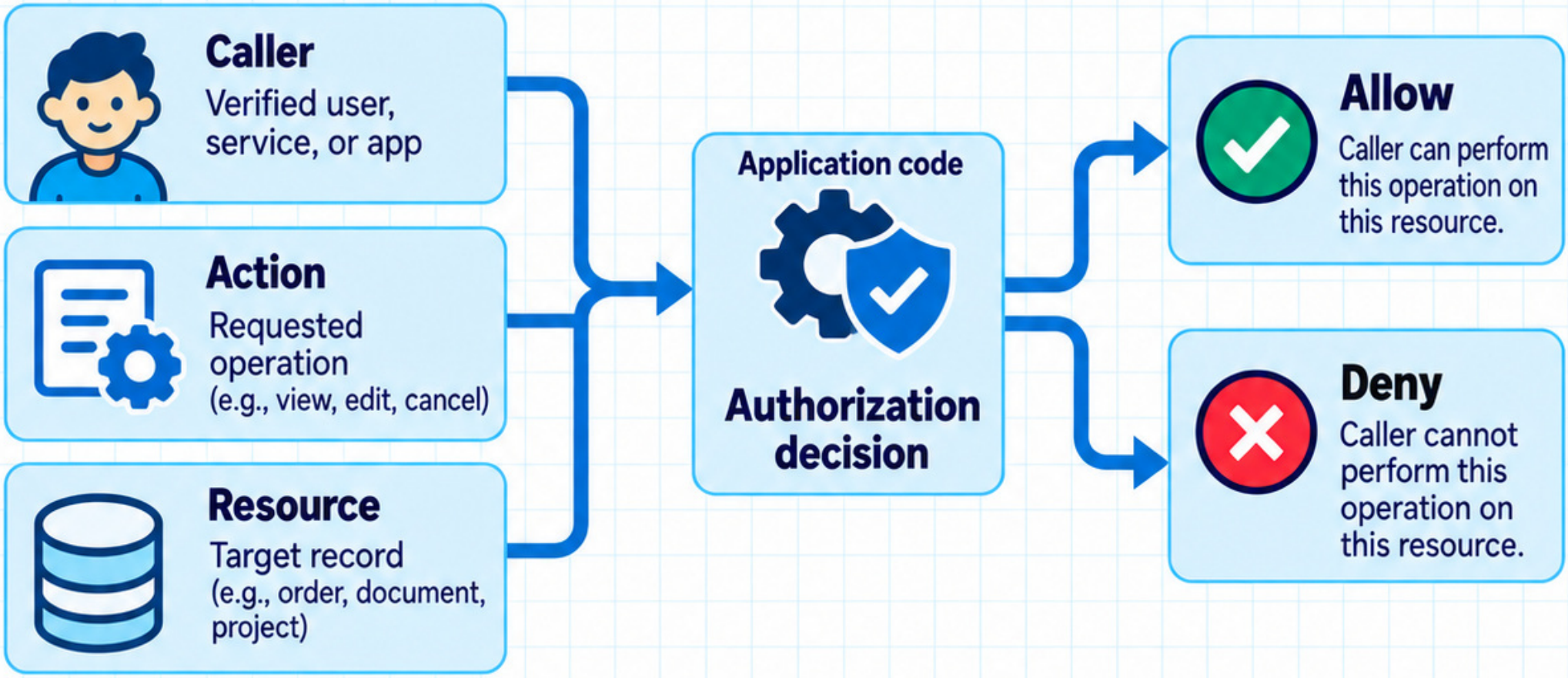


The application resolves that identity before it accesses protected data.

Text such as “I am the administrator” is part of the request, not proof of identity. A model’s interpretation of that text cannot replace credential validation.

AUTHORIZATION DECIDES THE PERMITTED OPERATION

Authorization asks whether this caller can perform this operation on this resource.



A signed-in customer can still lack access to a particular order.

The decision needs the verified caller, requested action, and target record. “The user logged in” is not a complete access-control rule.

BIND IDENTITY TO THE SERVER-SIDE REQUEST

The customer can supply an order ID.

The server obtains the customer identity from the verified session. It does not trust a customer ID invented by the browser or the model.

Session service

Verified identity
e.g. Maya

The business service checks whether that identity can read the requested order before returning protected fields.

Business service

Check access for
Maya + order #1042

Customer identity
(from session service)

Order ID
(from request)

Server

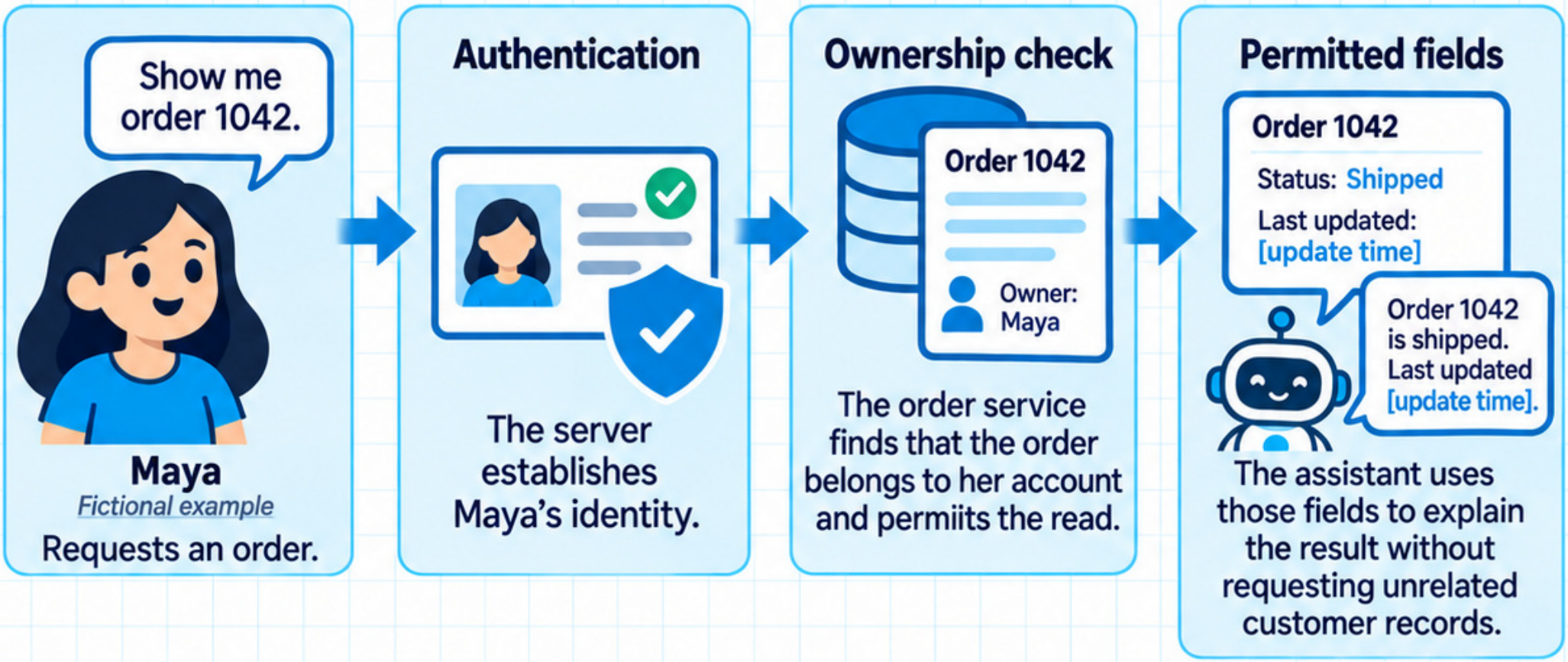
Resource check

Order data
(if authorized)

Fictional example

WALK THROUGH AN ALLOWED REQUEST

Maya asks for order 1042.

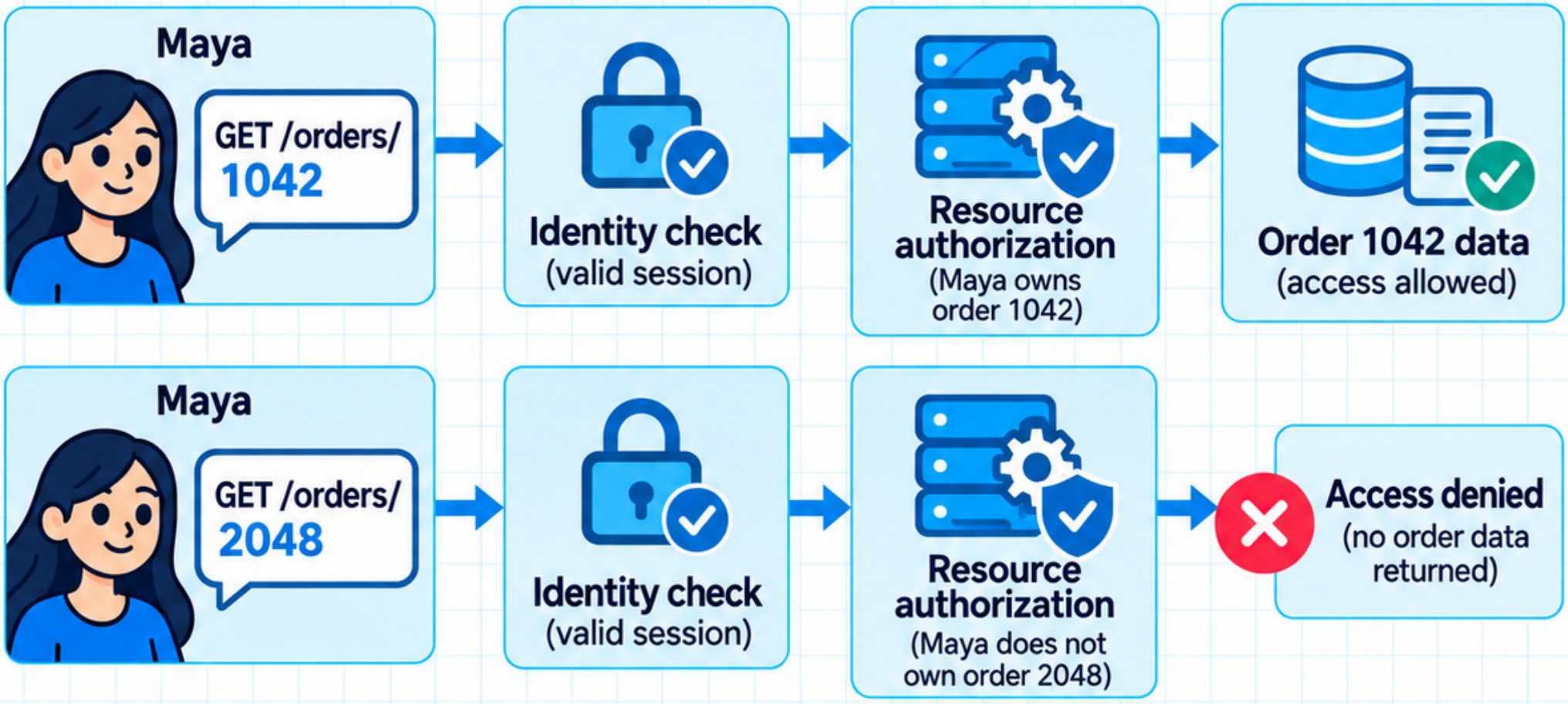


The server establishes Maya's identity. The order service finds that the order belongs to her account and permits the read.

It returns the status and update time. The assistant uses those fields to explain the result without requesting unrelated customer records.

NOW CHANGE ONLY THE ORDER ID

Maya changes the request to order 2048, which belongs to another customer.



The input still has a valid shape. The session is still valid. The resource check must deny access.

This test exposes a common design gap: checking login while forgetting permission for the requested object.

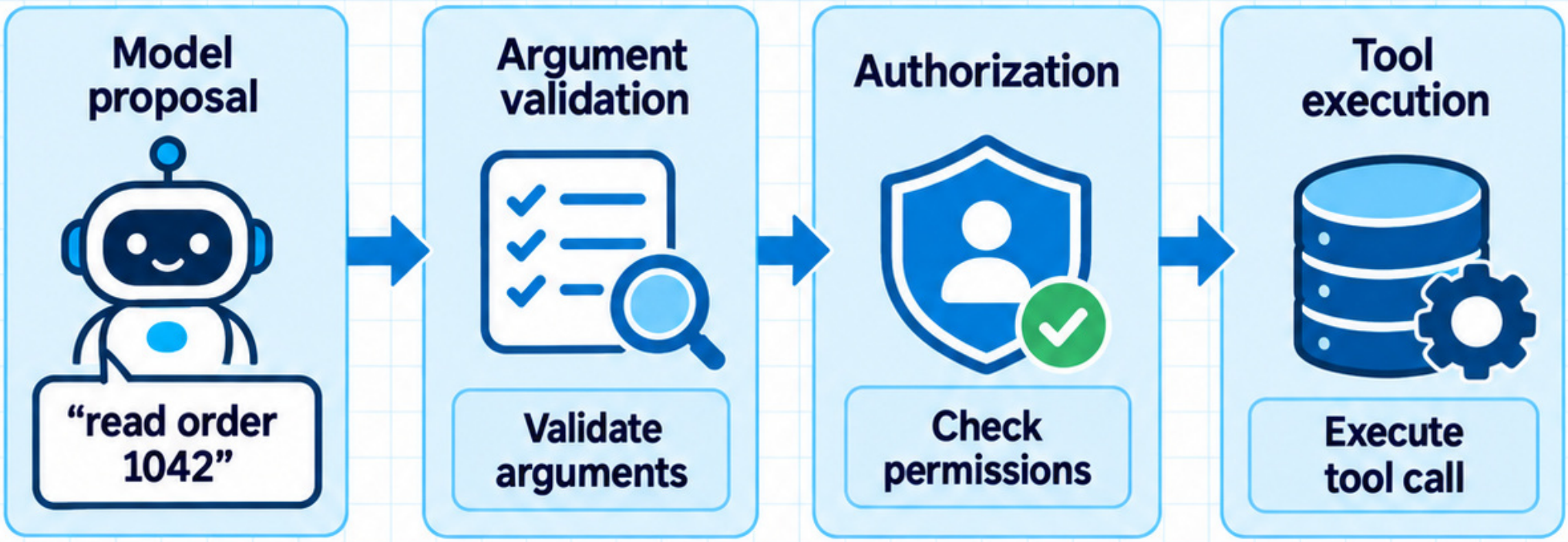
Fictional example

AI SYSTEM DESIGN • DAY 5

10/18

KEEP THE MODEL INSIDE THE BOUNDARY

The model can propose a tool call such as “read order 1042.”



Application code validates the arguments and checks authorization before execution.

A tool description or system prompt does not enforce access by itself. The business endpoint must apply its rules whenever a caller requests the operation.

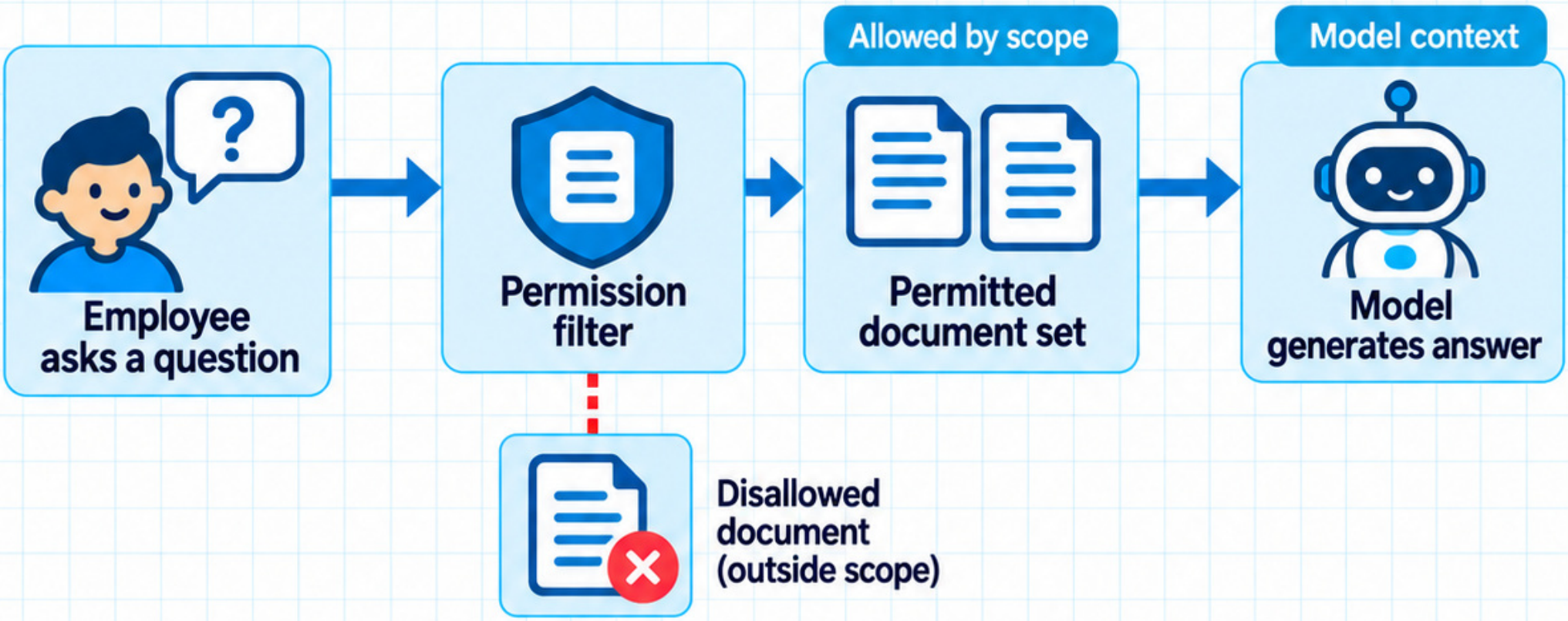
AI SYSTEM DESIGN • DAY 5

11/18

FILTER EVIDENCE BEFORE GENERATION

The same principle applies to document retrieval.

An employee can ask about a policy without permission to read every internal document.

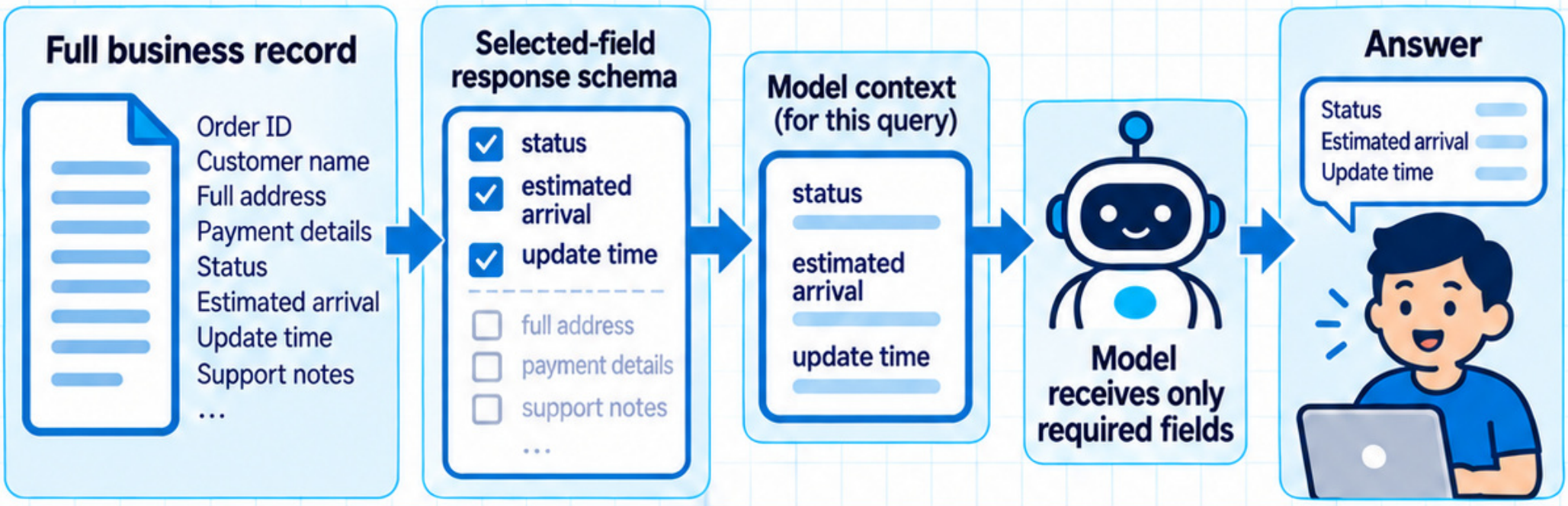


Retrieval must use the permitted scope before evidence enters model context. Hiding a source link after generation does not remove information the model already received.

@learn.machinelearning

RETURN ONLY THE REQUIRED FIELDS

An order-status answer can need status, estimated arrival, and update time.



It does not automatically need a full address, payment details, or every support note.

A narrow response contract reduces unnecessary exposure and simplifies downstream handling. Access to a record does not require returning every field in it.

SEPARATE USER IDENTITY FROM SERVICE CREDENTIALS

The customer identity tells the application whose request it is handling.

A service credential lets the application call a provider or backend service.

Customer (session)

User identity (e.g., user ID, token)

Application Service (your system)

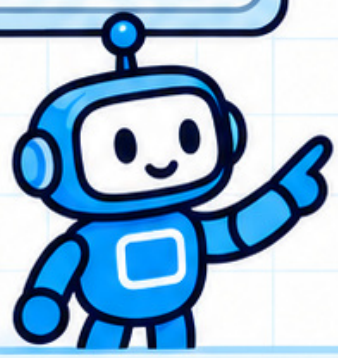
Identity verification (authenticate user)

Request authorization (enforce policy)

Service credential (stored in server configuration)

Model Provider (or backend service)

Receives service credential, not a claim of customer ownership.



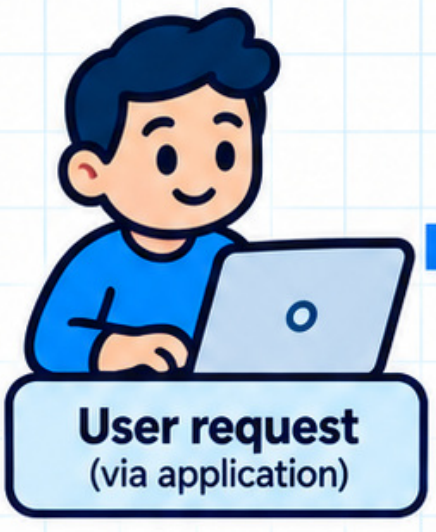
DO NOT include service secrets in the model prompt or context.

Service secret is applied by server authentication transport.

These identities serve different purposes. Keep credentials in controlled server configuration, and carry only the needed caller context across each business-service boundary.

DATABASE RULES ADD ANOTHER ENFORCEMENT POINT

PostgreSQL can restrict which rows a role can read or change.



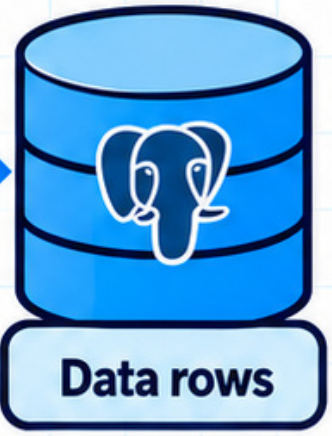
User request
(via application)

APPLICATION CHECK

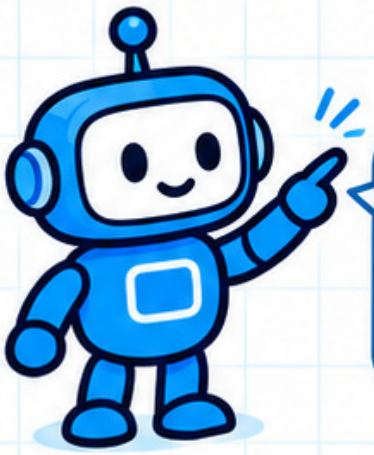
Application enforces its own authorization checks.

DATABASE ROW POLICY

PostgreSQL applies row policies to restrict which rows the role can read or change.



Data rows



Test with the real serving role to see what it can access.

Privileged-role exception

Owners and privileged roles can bypass ordinary row policies.

These policies can support application-level checks, but configuration matters. Owners and privileged roles can bypass ordinary row policies.

Use the actual serving role in access tests. A test run as an administrator can give misleading results.

DEFINE ERRORS WITHOUT EXPOSING EXTRA DATA

Invalid input
explain the request problem.

POST /orders

```
{
  "quantity": -1
}
```

```
{
  "error": "invalid_input",
  "message": "quantity
    must be a positive
    integer"
}
```

Unverified caller
require valid authentication.

GET /orders

```
Authorization:
(missing)
```

```
{
  "error": "unauthorized",
  "message": "valid
    authentication
    required"
}
```

Denied resource
return no protected fields.

GET /orders/123

```
Authorization:
 Bearer user_abc
```

```
{
  "error": "forbidden",
  "message": "access
    denied"
}
```

Unavailable service
return the defined failure response.

GET /orders

```
Authorization:
 Bearer user_abc
```

```
{
  "error": "service
    unavailable",
  "message": "please
    try again later"
}
```

Choose whether a denied response reveals that a resource exists. Keep that behavior consistent with the application's access policy.

CHOOSE TOOLS BY THEIR RESPONSIBILITY

FastAPI

FastAPI: receive requests and resolve dependencies.

Pydantic

Pydantic: validate input and output structures.

Identity service

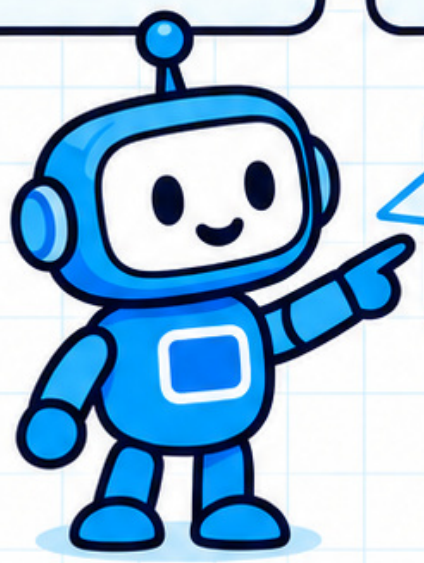
Identity service: validate credentials and establish the caller.

Business service

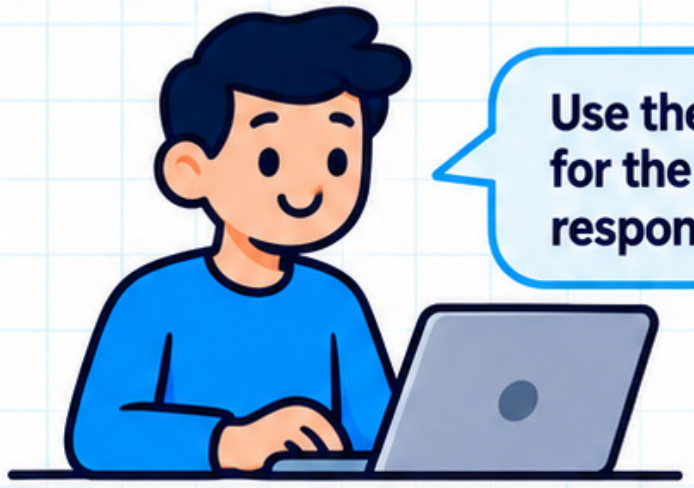
Business service: enforce resource and action permissions.

PostgreSQL

PostgreSQL: store records and apply configured data rules.



Each tool has a specific job in the request path.



Use the right tool for the right responsibility.

No single tool makes the whole path secure automatically.

TEST BOTH PERMISSION AND DENIAL

Create cases for:

- Correct owner
- Another customer's order
- Missing or invalid session
- A changed customer ID
- A disallowed operation
- A protected document in retrieval

Expected outcomes

Case	Expected outcome
Correct owner	Permitted fields
Other customer	Deny; no data
Invalid session	Authentication required
Changed identity	Cannot override caller
Disallowed operation	Deny operation
Protected document	Exclude from context

Inspect every stage

Request | Tool result | Retrieved context | Final answer

Check the final answer and every intermediate result.
A clean-looking answer can still follow an earlier data leak.

YOUR TURN

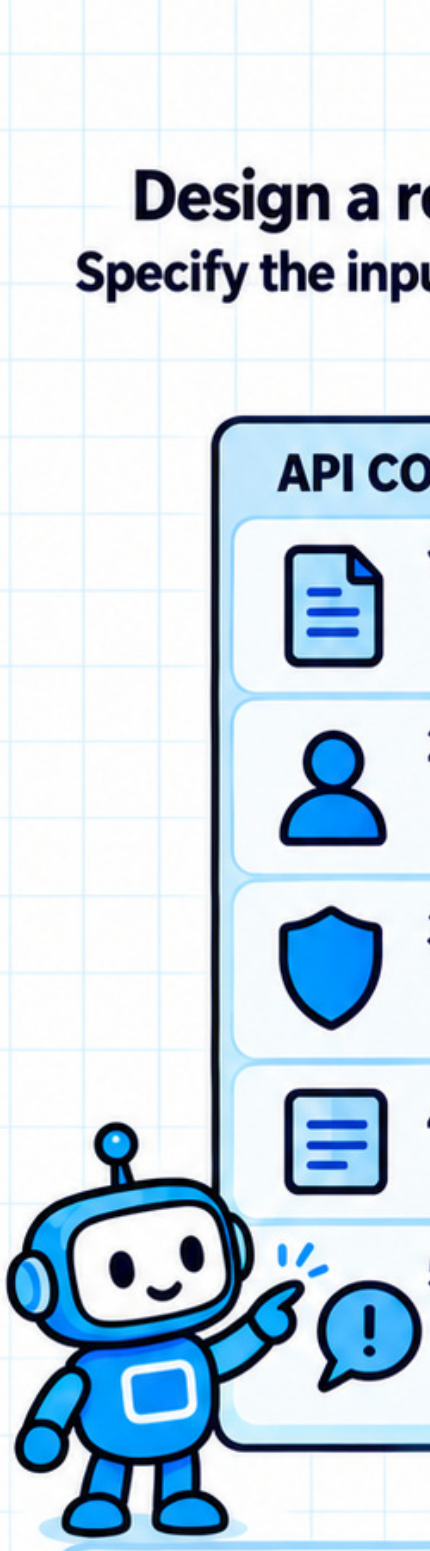
Design a read-order API or a demand-forecast API for two stores.

Specify the input, verified caller, permission rule, allowed fields, and denial response.

Add one test that changes only the resource ID.

API CONTRACT TEMPLATE

1. **Input**
   What parameters are provided?
2. **Verified caller**
   Who is calling and how is it verified?
3. **Permission rule**
   What must be true to allow access?
4. **Allowed fields**
   Which fields can be returned?
5. **Denial response**
   What is returned when access is denied?



REQUEST A

```
GET /v1/forecasts
?store_id=A
```

Caller: analyst_A
Resource: Store A

ENFORCEMENT GATES

Identity
Is the caller verified?

Authorization
Is the resource allowed?

ALLOWED
Return only allowed fields.

REQUEST B

```
GET /v1/forecasts
?store_id=B
```

Caller: analyst_A
Resource: Store B

ENFORCEMENT GATES

Identity
Is the caller verified?

Authorization
Is the resource allowed?

DENIED
Return denial response.

Fictional example: analyst_A may read Store A only.

Identity, shape, and permission answer different questions.

DAY 6: The evaluation contract