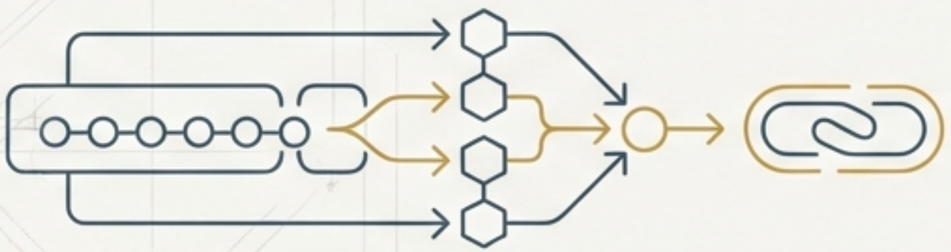
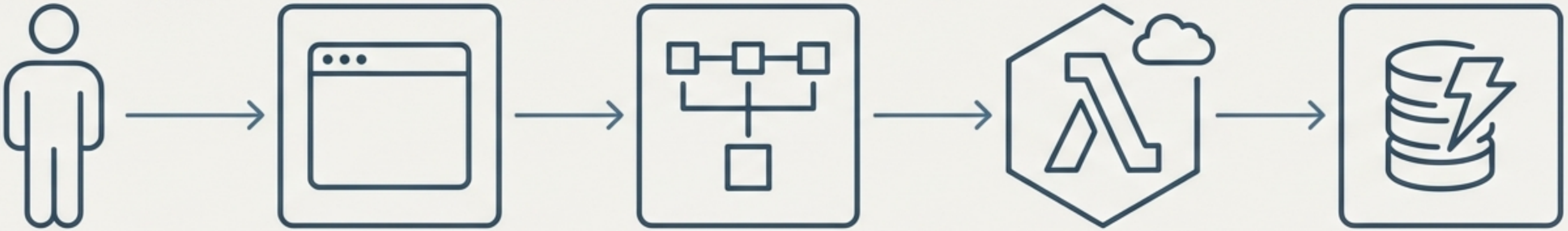
An Architecture Forged by Constraints

A Technical Deep Dive into a High-Performance, Serverless URL Shortener



The System at a Glance



A serverless architecture designed for speed and scalability, adapted specifically for the AWS Academy Learner Lab environment.

Part 1: The Ground Rules

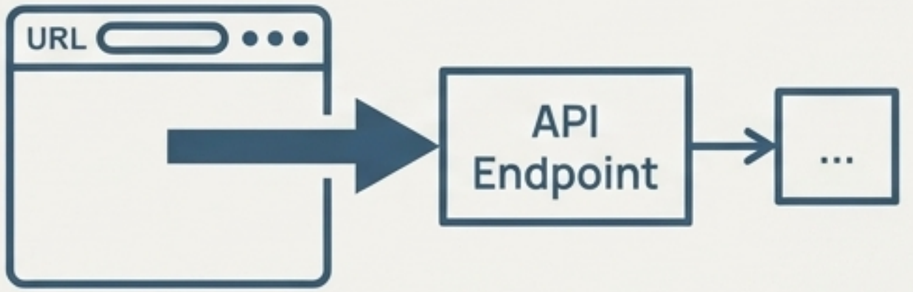
Engineering Within the AWS Academy Lab

Constraint #1: Adapting to Network & Hosting Limitations

Standard Practice

CloudFront for Edge Caching & Hosting

Typically, an S3 bucket hosts the Flutter web build, and CloudFront serves as a CDN to cache both the frontend assets and API responses.



Our Adapted Approach

Direct API Gateway Integration

The Flutter app communicates directly with API Gateway. This simplifies deployment and cache invalidation within the Academy environment, where S3 public website hosting is often restricted.

Constraint #2: Replicating Edge Logic for Geo-Location

Standard Practice

Leverage CloudFront Headers

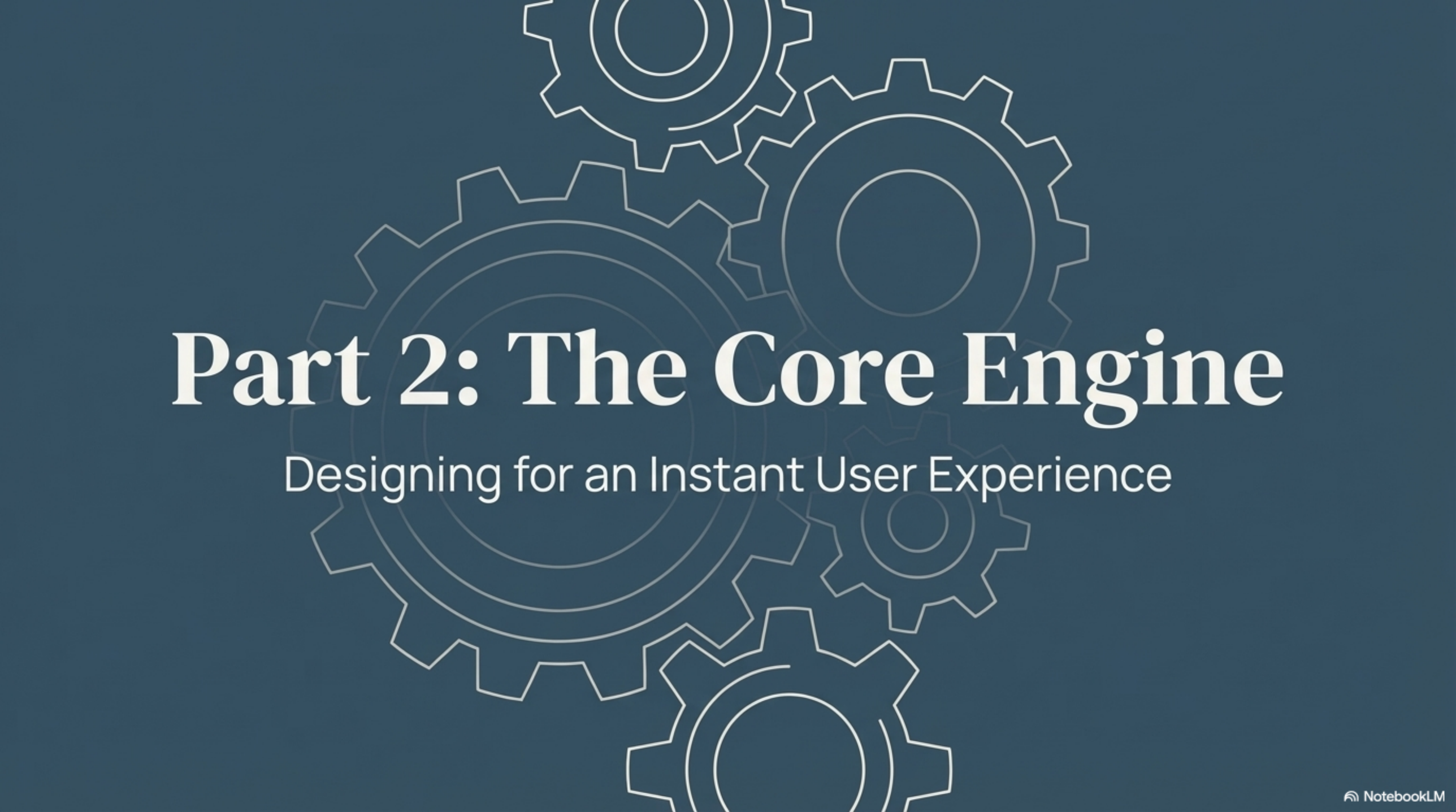
The `CloudFront-Viewer-Country` header provides a simple, low-latency way to identify a user's country at the edge.

Our Adapted Approach

Real-time Lambda IP Lookup

Since we bypass CloudFront, the Lambda function performs a real-time lookup against `ip-api.com`.

A **400ms timeout circuit breaker** is implemented to ensure that a slow third-party API never compromises the core redirect speed.



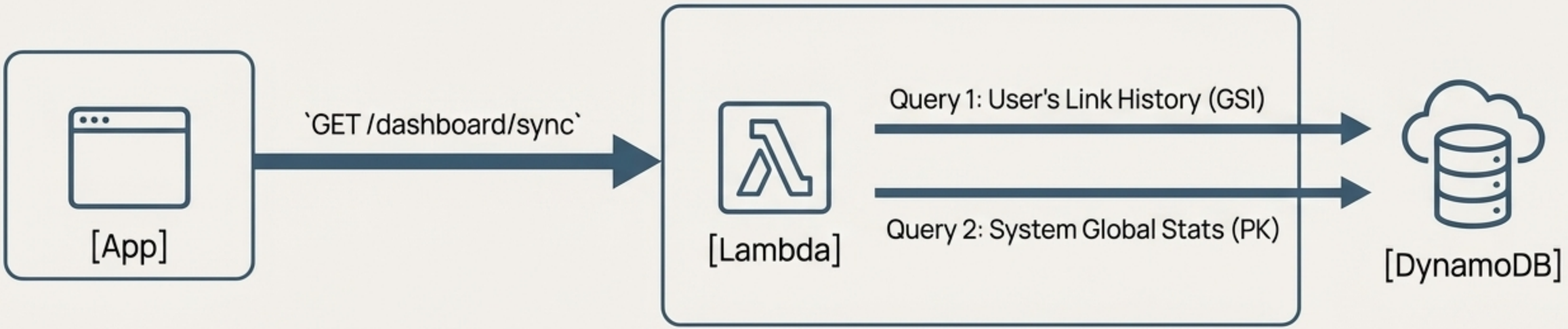
Part 2: The Core Engine

Designing for an Instant User Experience

The ‘Mega-Sync’ Engine: Eliminating a ‘Chatty API’

Multiple, sequential API calls for dashboard data create a slow, stuttering user experience.

The Flutter app makes **ONE** call to `GET /dashboard/sync`.

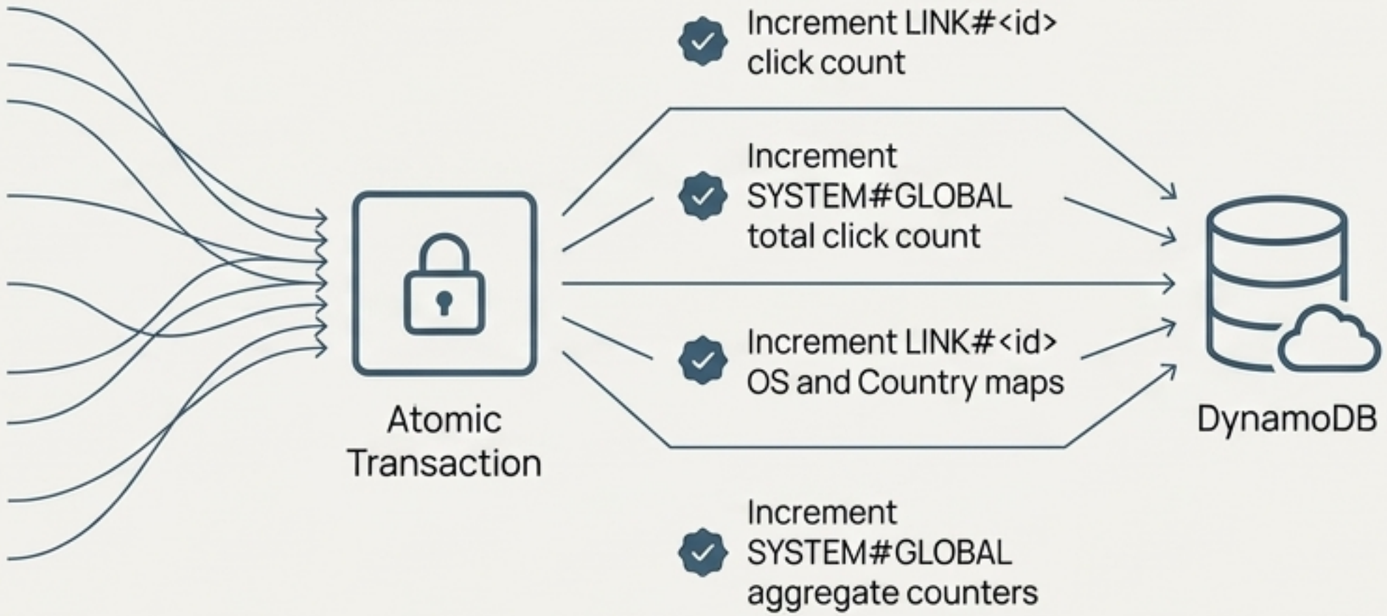


The UI renders the entire application state in a single frame update.

Atomic Analytics: Ensuring Bulletproof Data Consistency

A viral link can generate thousands of concurrent clicks, creating race conditions that corrupt analytics data.

DynamoDB `**TransactWriteItems**` or `UpdateCommand` ensures all updates for a single click event succeed or fail together.



All stats stay perfectly synchronized, no matter the load.

Smart URL Handling for Maximum Compatibility

The system minimizes interference with user input.

The backend only prepends `https://` if **no protocol** is detected.
The final `Location` header is sent exactly as stored.

User Input	Stored & Redirected As
`reddit.com`	`https://reddit.com`
`http://site.com`	`http://site.com`
`ftp://files`	`ftp://files`

Part 3: The Secure Perimeter

A Zero-Trust Approach to Authentication & Data Isolation

The Authentication Flow: Verifying Identity from Edge to Compute

1

Login

Flutter app uses the Secure Remote Password (SRP) protocol with an Amazon Cognito User Pool.

2

Token

Cognito issues a JSON Web Token (JWT).

3

Transport

The JWT is sent in the `Authorization: Bearer` header of every API request.

4

Verification

API Gateway natively verifies the JWT signature before invoking the Lambda.

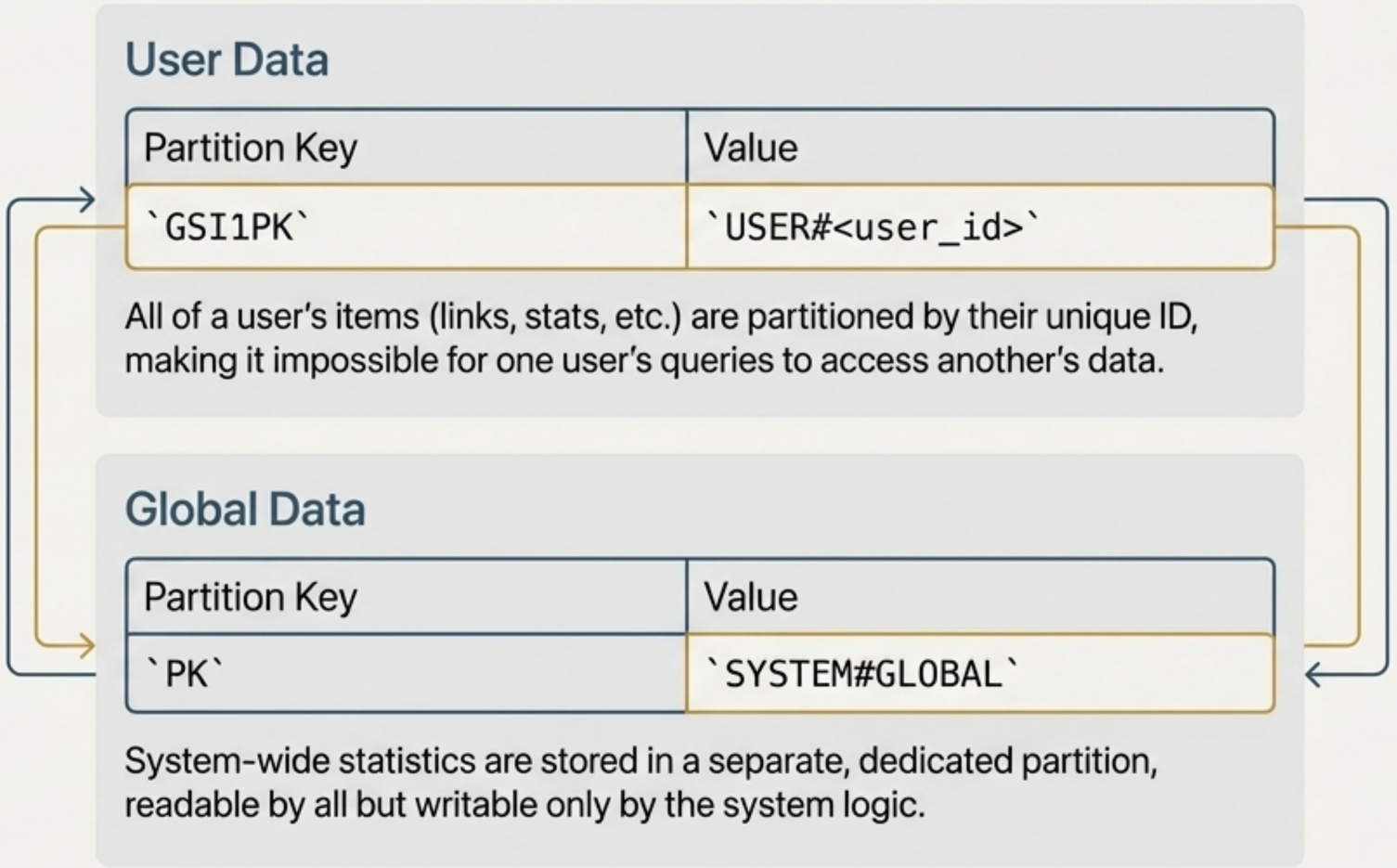
5

Context

The Lambda function securely identifies the user via the `event.requestContext.authorizer.jwt.claims.sub` field.

Enforcing Strict Data Isolation in DynamoDB

A single-table design with strategic partitioning to prevent data leakage between users.



Part 4: The Performance Profile

Delivering Speed and On-Demand Scale

The Results: Fast, Scalable, and Efficient

~150ms

Dashboard Load Time

Achieved via the 'Mega-Sync' single-query pattern.

~200-500ms

P90 Redirect Latency

Varies with the speed of the external Geo-IP lookup.

Horizontal Scale

Concurrency Handling

AWS Lambda and DynamoDB On-Demand seamlessly handle traffic bursts without manual intervention.

Thoughtful Design Triumphs Over Constraints

This serverless architecture delivers exceptional performance, consistency, and security. By embracing the specific limitations of its environment, we engineered targeted solutions like the ‘Mega-Sync’ engine and ‘Atomic Analytics’ that outperform standard patterns while remaining operationally simple.