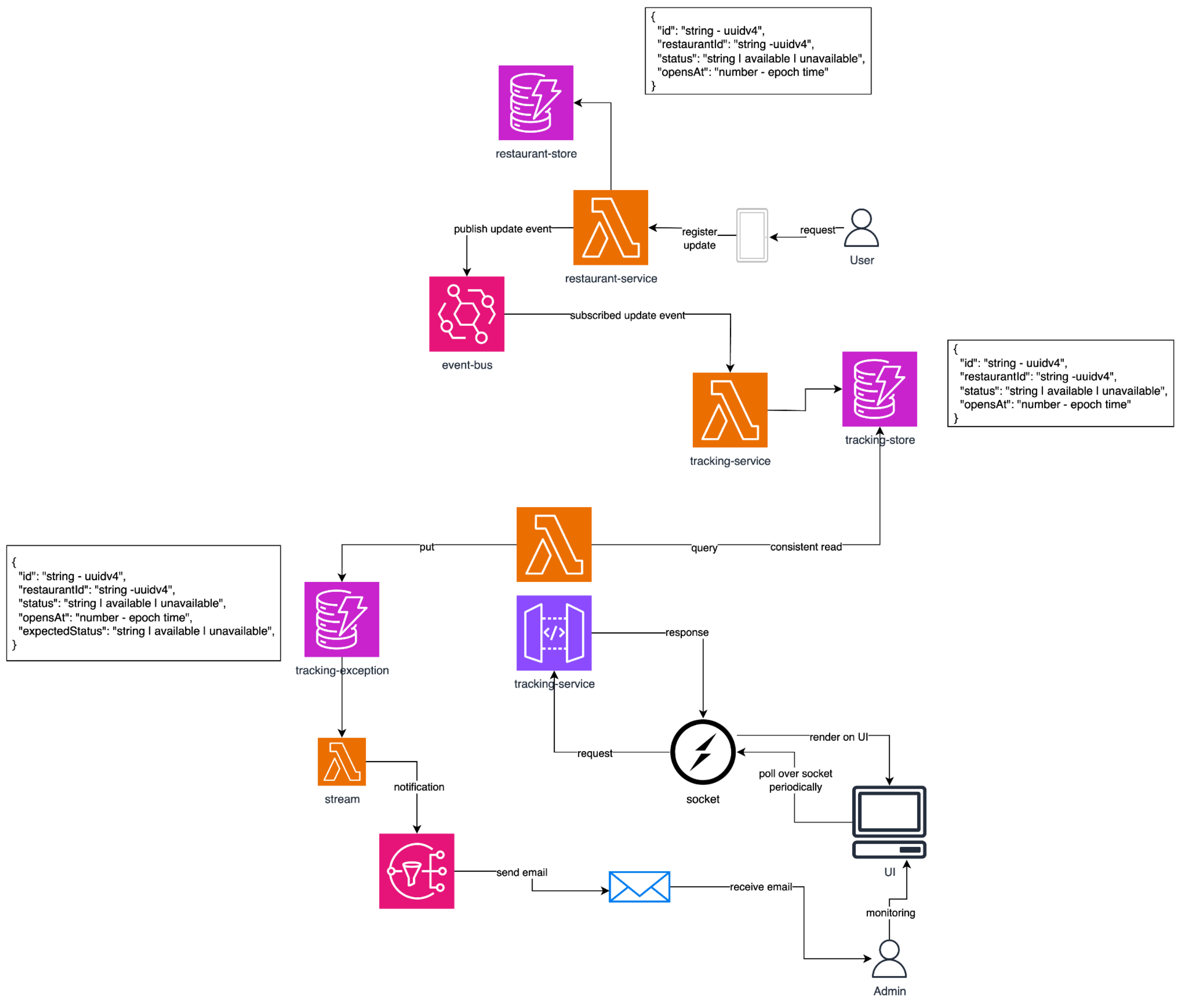
Purpose

This document covers the Architecture Document of a System that is built for Restaurant Chain Admins to track the availability of different Restaurant Listings under their chain and receive alerts for any discrepancies.

Architecture



```
{
  "id": "string - uuidv4",
  "restaurantId": "string -uuidv4",
  "status": "string l available l unavailable",
  "opensAt": "number - epoch time"
}
```

```
{
  "id": "string - uuidv4",
  "restaurantId": "string -uuidv4",
  "status": "string l available l unavailable",
  "opensAt": "number - epoch time"
}
```

```
{
  "id": "string - uuidv4",
  "restaurantId": "string -uuidv4",
  "status": "string l available l unavailable",
  "opensAt": "number - epoch time",
  "expectedStatus": "string l available l unavailable",
}
```

Service Layers

Restaurant Service : The responsibility of this service is to capture the inputs from the Restaurant Staff with regards to the availability of the Restaurant at a particular instance of time.

Tracking Service : The responsibility of this service is to consume the input in the form of an event, store the input and expose an API endpoint to consume the Restaurant Listing Info and also log exception

Resilience of the System

- Having multiple microservices would make the system have better availability and avoid single point of failure
- Having multiple data sources (redundant) will enable better data back-up & recovery
- Event Based mechanism will ensure delivery
- Use of serverless based services to avoid Ops

Data Concurrency at Scale

- Having consistent reads would ensure fetching of most up-to-date data
- Use of optimistic concurrency will prevent inconsistencies
- Event based delivery and writes to the DB would enable better concurrency control

Data Concurrency at Scale

- Write exceptions into an exception Sink
- Execute notification process for each exception
- Attach subscriptions like email, etc for better reception

Cost

- Polling Lambda Cost - $0 (negligible)
- API Gateway Cost - $0 (negligible)
- Lambda Stream Cost - $0 (negligible)
- 2 Dynamo DB (Write Only) Cost - $0.625 x 2
- Dynamo DB (Read + Write) Cost - 0.750
- Total Dynamo DB Cost = $2.00

- Total Restaurants - 10k
- Average Listing - 10 per Restaurant
- Requests per day per Restaurant = 10

- Requests = 10 x 10k x 10 = 1M
- 2 Lambdas Cost = $0.2 x 2 = $0.4

- Event Bridge (Publish) = 1M = $1.00

- SNS Cost = $0 (negligible)

- Total Cost = $3.40 per day

Implemented solution extension based on Architecture

- The simulated datasource (JSON based server) can be replaced by the Restaurant Service Layer
- Between the Web Client and the Restaurant Service Layer, the Tracking Service Layer can be introduced additionally to abstract the Data from Restaurant Service Layer to be used for either reporting on UI or to report the exception via Notification Layer
- UI can be extended to use persistent socket connection to be able to poll periodically for the Restaurant Listing Data for reporting
- Notification Layer can be augmented independently to the implemented solution