CS6310 – Software Architecture & Design
Assignment #4: Grocery Express Project – Architecture & Design (v1)
April 1, 2022
Team: Naren Bellamkonda, Yanlin Chen, Osvaldo Huerta Guzman, Xiaoyan Wang, Christine Manasterski

ADR – Scalability

TITLE
System Behavior Changes – Drone-Order Assignment

STATUS
Accepted

CONTEXT
Currently a drone is assigned to an order when the order is created. Then the user can add line items to the order only if the drone can carry those items. As the number of users grows, a lot of users might create new orders at the same time, these orders will be assigned to the same drone, causing the drone to run out of capacity quickly.

Example:

- Drone A - Capacity 100 lb.
- X number of users create 10 orders which are assigned to drone A.
- Problem scenarios:
    - 1 order exhausts all drone capacity. The other 9 orders will have to be canceled because the drone can't carry more items.
    - Orders exhaust drone capacity equally, but users want to add more items. This forces the users to either cancel the order or create a new order for the rest of the items.

DECISION
Assign an order to a drone when the order is ready to be delivered. Moreover, add a zip code to the system or any other location reference to load drones with orders that are close to each other.

CONSEQUENCES
The system will be able to manage a large number of orders. Customers will be able to add items to their orders without depending on the orders of other customers.

TITLE
Architecture / Infrastructure

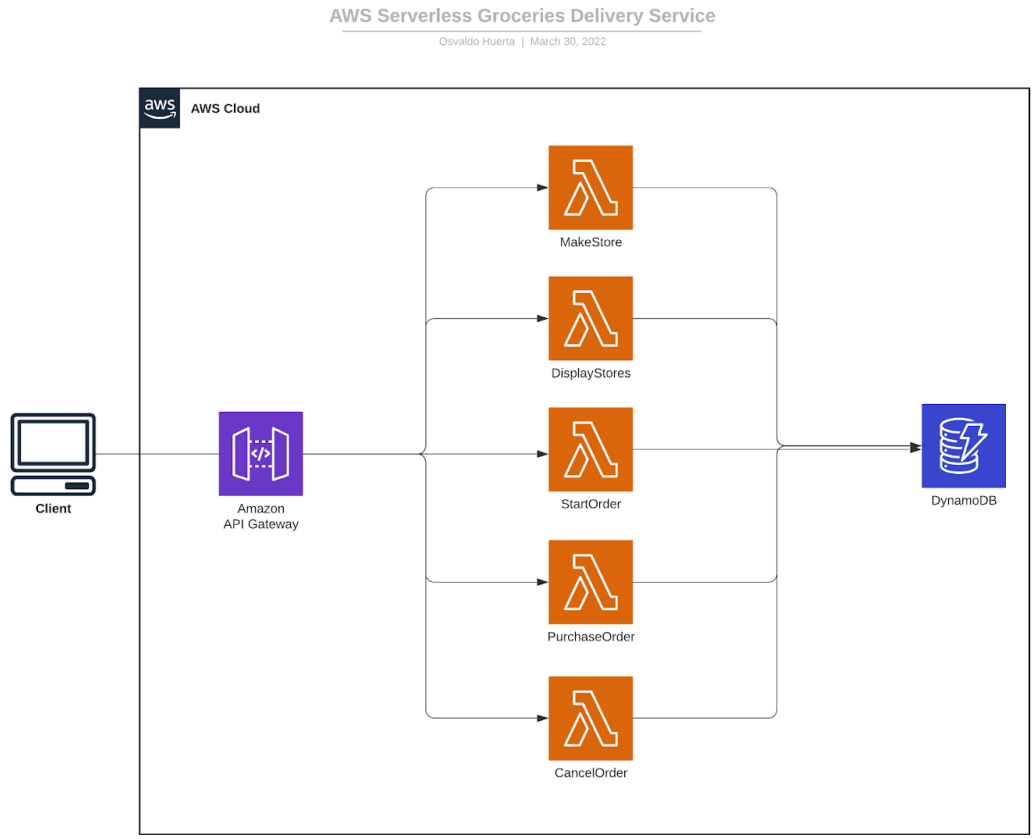
STATUS
Accepted

CONTEXT
The system should be able to operate and perform as the number of users and requests increases. Moreover, the system might be deployed across multiple regions in a large geographical area.

DECISION
We will build a serverless architecture to make the system scalable and easily deploy it across multiple regions.

Architecture Overview for Scalability:

- AWS Lambdas:
  - 1 lambda function per system operation (makeStore, displayStores, startOrder, etc).
- API Gateway:
  - 1 API Gateway to route system operations and trigger each lambda.
- DynamoDB (On-Demand capacity):
  - Tables:
    - Stores
    - DronePilots
    - PilotLicences
    - Customers
    - Catalog
    - Drones (Optional) (Could live inside Stores)
    - Orders (Optional) (Could live inside Stores)
    - Lines (Optional) (Could live inside Orders)



CONSEQUENCES
A serverless architecture will allow us to handle availability and scalability. The system will be easy to deploy to a large geographical area. Moreover, it will reduce risk to our cost from over-provisioning and reduce risk to our performance from under-provisioning.

NOTES
Author: Osvaldo Huerta
Approved By: Group 80 Members, 03/31/2022
Last Updated: 03/31/2022 by Osvaldo Huerta

ADR – Authentication & Authorization
TITLE
Authentication & Authorization Design Considerations

STATUS
Accepted

CONTEXT
Authentication and authorization are often essential to enforce any security and/or privacy protocols. Building in authentication for grocery delivery is important because it creates a layer of restriction to

prevent bad actors from gaining access to the application. Authentication refers to security requirements to ensure users are who they say they are. Authorization refers to security requirements to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule, field level, etc.).

DECISION

All login attempts will be authenticated by a user id and password. Login id authentication can be a phone number or combination of firstname and lastname. We will need to include an indication as to the user level role, such as whether the user is a customer or an employee to indicate the business role they address and configure the accesses as appropriate to the user. These roles should be able to be created, updated, or deleted. Any new users with no role should not have access. We should be able to specify or change user permissions, such as read and write.

Multi-Factor authentication (MFA) can enhance security at the user level to defend against account hacks. We need to provide a workaround in the event that users lose their phones or SIM cards which they would use to generate a new authentication code. Token-based authentication enables users to enter their credentials once and receive a unique encrypted string of random characters in exchange. They can then use the token to access protected systems instead of re-entering credentials. The digital token proves that the user already has access permission. Use cases of token-based authentication include RESTful APIs that are used by multiple frameworks and clients. When the user shows up at the authentication gateway, the same authentication gateway can instruct firewalls to enable the desired IP address.

CONSEQUENCES

If a new user with no defined role is granted access to the system, they may be able to collect, contaminate, or otherwise have an adverse effect on sensitive information. Therefore it is crucial to mitigate any risks of unauthorized or unauthenticated access to the system with strict roles defined that engage with the security requirements as well to ensure an end to end experience that protects sensitive user and company information.

COMPLIANCE

It is important to ensure that any user information that is stored is done so in full compliance with global data protection laws (such as GDPR) for all geographical locations in which the company and its users operate and reside. Failure to do so may result in fines or other legal action. User should also have the ability to request that their data is removed from the system per all local guidelines. We would want to ensure compliance with these laws by subjecting designs to a comprehensive review by a legal team.

NOTES

Author: Christine Manasterski
Approved By: Group 80 Members, 04/01/2022
Last Updated: 04/01/2022 by Christine Manasterski

ADR – Security (Assigned)

TITLE
Encrypting sensitive personal information at rest and transit to comply with Federal privacy laws

STATUS
Accepted

CONTEXT
Functional requirement mandates to store the customer personal information and also employee personal information for the payroll benefits and performance reasons.
To support this requirement, during the registration/singing up the account, customer were asked about their home address, phone number and email address. This information will be saved in the customer reference tables that will used during the order creation and future customer support purposes.
Employee were asked to provide their Taxid and Pilot License ID part of the employment registration process. All this information will be used for running the payroll and providing the additional benefits part of the employment

This data being SPI & PII as per the privacy laws needs to be encrypted at rest and transit. To address encryption need to completely understand benefits and consequences on performance of the application holistically. When we encrypt this data at some point to transmit to downstream service this data needs to be decrypted. And also when encrypted in disk/storage this data needs to be encrypted in memory(RAM cache) as well to avoid in case there is a system crash, when the application performs the data dump, all these data cannot be on readable format.

DECISION
We will encrypt the SPI data related to employees and customer and to encrypt data we can use native Java libraries which is Java Cryptography Extension (JCE)

Assumption is that only SPI/PII data is encrypted, other will be a open text

When Customer and Employee enters the sensitive personal information, method should call the encryption method to encrypt data in the java memory address space and later write the data to database
And all this data should never be logged in any of the application logs even if use the debug mode for logging

CONSEQUENCES
Every organization collects, stores and uses PII/SPI data of their employees/customers for the business process needs. For running the payroll organization needs SSN of the employee and similar other data elements to support business function needs.
So part of this it becomes the organization responsibility to protect and secure sensitive information to avoid a risk of a breach and not meeting compliance standards.

COMPLIANCE
Need to perform periodic key rotation process to make sure, in case there is a breach, we immediately change the cipher key.
Store the private key in the vault that can be accessed by only authorized IT compliance team
Process is properly documented and approved by compliance team.

NOTES
Author: Naren Bellamkonda
Approved By: Group 80 Members, 03/28/2022
Last Updated: 03/29/2022 by Naren Bellamkonda

Planned Technical Deployment

STATUS
Accepted

CONTEXT
The tech stack for this project must be scalable, effective distribution of load to meet the future demand, but at the same time, we don't want to invest too much infrastructure at the beginning. As a delivery application, it's designed for easy onboarding and searchability of grocery stores. Prediction of the delivery time for order is under account. It must allocate and monitor orders for drone pilots effectively and efficiently, and the ability for Service Providers to send notifications to customers, drone pilots, stores. We'll have a configurable database/s and data access layer to yield high performance and throughput for users and employees.

DECISION
[Server and System] The project will be based on a serverless CI/CD (continuous integration and continuous delivery) environment on the Amazon Web Services (AWS) Cloud. We'll set up application scaling across multiple services using AWS Auto Scaling.
[Database] We'll launch a fully managed Amazon DynamoDB NoSQL database.
[API Gateways] All order requests made from a mobile app or UI will go to different services via the API gateway.

- Payment gateway integration: PayPal, Stripe, and Braintree
- Realtime notification for customers and Pilot: Amazon SES
- GPS map integration: Google Map API
- Order/Inventory Distribution: Apache Kafka (a highly scalable and open-source distributed event store and stream-processing platform, for handling real-time data feeds, for Inventory/Order distribution)

[Language] Java, SQL, Swift, Node.js, JavaScript, HTML
[Authentication] We'll use Amazon Cognito for authentication, authorization, and user management for web and mobile apps.

CONSEQUENCES

[Server and System] By using AWS serveless CI/CD, we're able to collaborate securely and efficiently on serverless application deployments as well as other resources on AWS. With Auto Scaling on AWS, we're able to monitor the applications and automatically adjust capacity to maintain steady, predictable performance.

[Database] AWS DynamoDB is highly scalable and provides support for JSON-styled, document-oriented storage systems. Its flexible data model enables you to store data of any structure, and it provides full index support, sharding, and replication.

[API Gateways] API gateway will take care of load balancing and routing requests to services.

[GPS map integration] For drones to deliver on-time, they need to choose the shortest and fastest routes to reach the user locations, which will also help them in reaching the correct destination based on the user requests.

[Streaming Data/Realtime] Kafka ensures exactly once delivery between producer and consumer applications through the newly introduced Transactional API.

[Language] Java is the main language we use in this project; we'll use SQL to manage the database. We'll develop both Android and IOS versions for the Mobile App using Java and Swift and use Node.js, JavaScript, HTML for the Web UI.

[Authentication] Users can sign in directly with a username and password, or through a third party such as Facebook, Amazon, Google or Apple.

COMPLIANCE

Our biggest challenge of lacking a solid algorithm for the time-of-delivery prediction model, we'd like it to be accurate as much and to be flexible enough to handle various scenarios due to its uniqueness of newly surfaced information in different stages.

We'll start with an internal version to test the capabilities of the architecture. In terms of cost, we'll use the billing on AWS and monitor our budget based on the performance, we'll adjust our plan accordingly. Kafka is open-source and free for now. We'll have to pay for third parties, like Payment and Google Map API.

NOTES

Author: Yanlin Chen

Approved By: Group 80 Members, 03/30/2022

Last Updated: 03/30/2022 by Yanlin Chen