**Computer Engineering Department**

**CX: Cloud Computing**

**COE 427 & COE 452**

**Term 221**

|  |  |
| --- | --- |
| **Project Title** | **Shipments Delivery System** |
| **Team Members**  **(Name + ID)** | Amer Mosally 201820680  Ahmed Alrimawi 201832320  Ali Abu Rub 201868360 |
| **Problem Statement** | E-commerce is booming right now. A key component of success is  managing logistics efficiently, and the least optimized part of this  process is customer delivery. Traditionally, drivers are assigned to  shipments in semi-random locations, causing delays. This proposal tries  to solve this problem by implementing a cloud-based solution that  stores the shipment information and runs algorithms to optimally  distribute shipments among couriers. |
| **Who Are Your Customers?** | * E-commerce businesses * Logistics businesses |
| **Value Proposition** | Businesses implementing the solution will have shorter delivery times  while increasing couriers’ daily capacity. In addition, shipments would  be processed faster in fulfillment centers and the customer will be satisfied. |
| **Services + API** | Decompose your system into services.   |  |  |  |  | | --- | --- | --- | --- | | **Service Name** | **Description** | **HTTP Method** | **Route** | | Courier Service | Insert the courier ID, name, and coordinates. | POST | /courier\_management | | Courier Service | Read all the registered couriers. | GET | /courier\_management | | Courier Service | Delete the courier from the database. | DELETE | /courier\_management | | Shipment Service | Insert the shipment ID, date, and coordinates. | POST | /shipment\_management | | Shipment Service | Read all the registered shipment. | GET | /shipment\_management | | Shipment Service | Delete the package from the database. | DELETE | /shipment\_management | | Manual Attach Service | Insert the package ID, courier ID, and the location coordinates. | POST | /manual\_attach | | Manual Attach Service | Read all packages with their associated courier. | GET | /manual\_attach | | Email Notification Service | Send email with the result database | GET | /email\_notification | |
| **Feedback on your Course project Proposal** | Approved  1. You need to add more services (as Lambdas) in the serverless part of the architecture.  2. Add one more microservice in the infrastructure (matching + scheduling). Use API gateway in the infrastructure.  uS1 = Matching (courier -> Shipment: maximize profit)  uS2: Scheduling (Courier -> stop points: minimize cost)  uS3: Add a third one |
| **Workload Characterization** | Describe the types of HTTP requests and their payloads (i.e., HTTP method and JSON files)   |  |  |  | | --- | --- | --- | | **HTTP Request** | **HTTP Method** | **JSON File** | | Insert the ID, name, and coordinates in the database. | POST | {  "ID":2018,  "name":"Amer",  "x":"2",  "y":"3",  “available”:”TRUE”  } | | Read all the courier names from the database. | GET |  | | Insert the Shipment details. | POST | {  "ID":2018,  "date":"2022/12/18",  "x":"2.66",  "y":"34.2"  } | | Read all the shipment from the database. | GET |  | | Insert courier and shipment details. | POST | {  "PackageID":2334018,  "CourierID":"4346",  "date":"2022/12/18",  "x":"3",  "y":"3.4"  } | | Triggers ec2 to assign drivers to packages with specified date | POST | {  "date": "2022/12/18"  } | | Read all items from the result database | GET |  | | Send email with result database (shipment with courier) | GET |  | |

A**rchitecture Diagram of your application.**

* Use a professional tool
* Use the standard AWS icons (if you are in COE 452)
* Draw the microservice architecture (this applies to both COE 452 & COE 427)
* Your terraform file (and Ansible file) should be compatible with the architecture diagram

