API Driven Past Paper

Q&A

Q.1 A currency calculator application takes "src_country", "src_currency", "dst_country", and "amount" as inputs to the service, and returns the "dst_currency" and "converted_amount" as outputs from the service. Design a gRPC proto file for this service. [6 marks]

A gRPC service is described using a .proto file, which specifies the request and response messages as well as the RPC methods. For a currency calculator, we need a single RPC call:

```
syntax = "proto3";
package currency;
service CurrencyCalculator {
  rpc ConvertCurrency (CurrencyRequest) returns (CurrencyResponse);
}
message CurrencyRequest {
  string src_country = 1;
  string src_currency = 2;
  string dst_country = 3;
  double amount = 4;
}
message CurrencyResponse {
  string dst_currency = 1;
  double converted_amount = 2;
}
```

Q.2 Challenge the accuracy of the below statements (agree/partially agree/disagree) with proper justification.

a. DevOps principles can be directly applied to Machine Learning projects. [2 marks]

Answer: *Partially Agree.*

DevOps deals with CI/CD, automated testing, and deployment pipelines. These principles are valuable for ML projects too, but machine learning adds new complexities such as handling changing data, retraining models, ensuring reproducibility, and monitoring model drift. Hence, the direct application of DevOps is not enough; they need to be extended into MLOps.

b. HTTP POST method is considered idempotent. [2 marks]

Answer: *Disagree.*

An **idempotent** HTTP method is one where making the same request multiple times has the same effect as making it once. For example:

- `GET /users/1` always retrieves the same resource.
- `PUT /users/1` with the same body will always replace the resource with the same state.

But `POST` is **not idempotent**, because each POST request can create a new resource or perform an action with different side effects. For example, posting the same order twice to `/orders` can create two separate orders.

c. Cloud-native applications are typically deployed more rapidly than cloud-based applications. [2 marks]

Answer: *Agree.*

Cloud-based applications are usually traditional monolithic systems that have simply been moved into the cloud, often running on virtual machines. While this provides cloud benefits such as managed infrastructure, deployment is still slower and more rigid because the architecture was not originally designed for automation or rapid scaling. In contrast, cloud-native applications are designed specifically for the cloud environment. They use microservices, containerisation, and orchestration platforms such as Kubernetes, along with CI/CD pipelines that support rolling updates and automated scaling. This makes deployment and iteration significantly faster compared to cloud-based applications.

Q.3. Below is a snapshot of a “lung cancer” dataset. The input variables include Gender, Age, Smoking, Anxiety,

Coughing and Chest_Pain. In the input variables, the value “1” indicates “yes” and value “0” indicates “no”. For example, a patient who smokes, has a value assigned as 1, and a non-smoker has a value assigned as 0. The output variable is “Lung_cancer”, where “Yes” indicates that the patient is having lung cancer, and “No” indicates otherwise.

a. Draw and briefly explain the generic machine learning life cycle. [3 marks]

b. Apply the machine learning life cycle on this dataset and discuss each phase of the lifecycle. [3 marks]

Gender	Age	Smoking	Anxiety	Coughing	Chest_Pain	Lung_Cancer
M	65	1	1	0	1	NO
F	55	1	0	1	0	NO
F	78	0	1	1	1	YES
M	60	0	1	0	0	YES
F	80	1	0	1	0	NO
F	58	1	1	0	0	YES
F	70	1	1	0	1	YES
F	74	0	1	1	1	NO
M	77	1	1	1	0	NO

a. Draw and briefly explain the generic machine learning life cycle. [3 marks]

The **ML lifecycle** has six main steps:

1. **Data Collection** – obtain relevant raw data.
2. **Data Preparation** – clean, encode categorical values, normalise numbers.
3. **Model Training** – select an algorithm and train it on data.
4. **Evaluation** – test the model with unseen data to measure accuracy, precision, recall, etc.
5. **Deployment** – integrate the model into a usable application.
6. **Monitoring** – check performance over time and retrain if accuracy drops.

b. Apply the machine learning life cycle on this dataset and discuss each phase. [3 marks]

Data Collection: We have patient records with variables like smoking, anxiety, coughing, etc.

- **Preprocessing:** Encode categorical fields like gender, normalise the age values, and check for missing entries.
- **Model Training:** Use a classification algorithm (e.g., Decision Tree or Logistic Regression) to learn the relationship between symptoms and lung cancer.
- **Evaluation:** Measure model performance on unseen rows of data, using metrics like recall (important for medical problems to reduce false negatives).
- **Deployment:** Build an API that takes patient data and predicts lung cancer risk.
- **Monitoring:** Continuously update the model as more patient data becomes available, retraining when accuracy drops due to new patterns in the population.

Q.4. As a software developer responsible for managing a “music database” API, you are tasked with implementing the following functionalities:

- Adding new albums to the database.
- Updating album details.
- Deleting an album.
- Retrieving information about a specific album.

Assume that you are starting with version 2.0.0 of the API, and the base endpoint is “/api/music”. The album model includes fields such as 'id', 'title', 'artist', 'releaseDate', and 'genre'. Design the API endpoints using RESTful HTTP verbs (GET, POST, PUT, DELETE), and show the request body where applicable. [6 marks]

Base endpoint: `/api/music/v2.0.0/albums`

- **Add Album (POST):**
  `POST /api/music/v2.0.0/albums`

```
{
  "title": "Thriller",
  "artist": "Michael Jackson",
  "releaseDate": "1982-11-30",
  "genre": "Pop"
}
```

- **Update Album (PUT):**
  `PUT /api/music/v2.0.0/albums/{id}`

```
{
  "title": "Thriller (Remastered)",
  "artist": "Michael Jackson",
  "releaseDate": "2001-10-16",
  "genre": "Pop"
}
```

- **Delete Album (DELETE):**
  `DELETE /api/music/v2.0.0/albums/{id}`

- **Retrieve Album (GET):**
  `GET /api/music/v2.0.0/albums/{id}`

Each endpoint uses the appropriate HTTP verb to clearly indicate intent.

Q.5 a. Differentiate between the terms “DataOps” and “MLOps”. [2 marks]

Answer:

- **DataOps:** Focused on automating and improving the reliability of data pipelines. It ensures timely, high-quality, validated data for analytics and downstream systems.
- **MLOps:** Extends DevOps to machine learning, focusing on model training, deployment, monitoring, and retraining. It ensures models are production-ready and adapt to changing data.

5 (b) Quick commerce applications focus on the rapid delivery of goods and services, often promising delivery times as short as 10 to 30 minutes. Examples include Swiggy Instamart, Zepto, Big Basket etc., Explain how you would apply DataOps and MLOps principles to enhance the performance and efficiency of these applications.? Provide specific examples. [4 marks]

- **DataOps in Quick Commerce:**
  - Automates the flow of order, inventory, and location data.
  - Ensures freshness and consistency of stock data so customers don't order items that are unavailable.
  - Example: BigBasket maintaining live stock counts across warehouses.
- **MLOps in Quick Commerce:**
  - Uses ML models for demand forecasting and delivery-time predictions.
  - Continuously monitors models and retrains them with new customer data, traffic patterns, and weather conditions.
  - Example: Swiggy Instamart retrains models daily to adapt to festival seasons where demand spikes.

DataOps guarantees clean and real-time data, while MLOps ensures adaptive, high-performing models. This combination directly improves delivery speed, efficiency, and customer satisfaction.