vrittech

Thank you for participating in the interview process.

Please find below the list of tasks assigned to you.

Deadline: Please complete the task within **5 days of receiving this email**.

Note:

- Candidates are encouraged to complete all tasks within the given timeframe, as late submissions will result in disqualification.
- Any form of external assistance, including the use of AI tools or seeking help online, will lead to disqualification.
- The tasks are designed to assess your **individual reasoning and problem-solving abilities**, so we expect independent effort.

Wishing you the best of luck!

Task A.

Travelling Salesman Problem

A salesman must visit 6 cities: A, B, C, D, E, F.

The distances (in kilometers) between the cities are given below:

From / To	**A**	**B**	**C**	**D**	**E**	**F**
A	**0**	**10**	**15**	**20**	**25**	**30**
B	**10**	**0**	**35**	**25**	**17**	**28**
C	**15**	**35**	**0**	**30**	**28**	**40**
D	**20**	**25**	**30**	**0**	**22**	**16**

E	**25**	**17**	**28**	**22**	**0**	**35**
F	**30**	**28**	**40**	**16**	**35**	**0**

Task:

The salesman must:

1. Start from City A.
2. Visit each city exactly once.
3. Return to City A.

👉 Question: What is the shortest possible route the salesman can take, and what is the total distance?

Task B.

As a Python developer, how would you **design and implement** a **scalable and efficient email validation tool** similar to services like **NeverBounce** or **ZeroBounce**?

The tool should:

1. Accept **multiple email addresses** as input via an **API**.
2. Check if the **email format** is valid.
3. Verify whether the email can **actually receive messages** using **DNS** and **SMTP validation**.
4. Fetch and report the status of **SPF**, **DKIM**, and **DMARC** records for the domain.
5. Expose an **API endpoint** that returns the results as an **array of objects** (or any efficient data structure you prefer).

Task:

- Propose the **best approach, architecture, and technologies** in Python to make the solution **highly scalable** and **efficient** when validating **thousands or millions of emails**.

- Then, **write a Python program** that demonstrates this solution, including:

    - Email validation logic
    - DNS/SPF/DMARC/DKIM checks
    - SMTP verification.

- A simple REST API endpoint to return the results (using Django).