# Embassy Consulate Line System Report

Rupam Debnath Rodra

23-51418-1

## 1. Understanding the Problem

The Embassy Consulate Line System is designed to streamline the process of handling visa applicants at an embassy. Applicants for different visa types — Tourist, Medical, Business, and Government Officials — are served at four dedicated counters. Each counter gives priority to its assigned visa type but can also serve applicants from the longest available queue if its primary queue is empty.

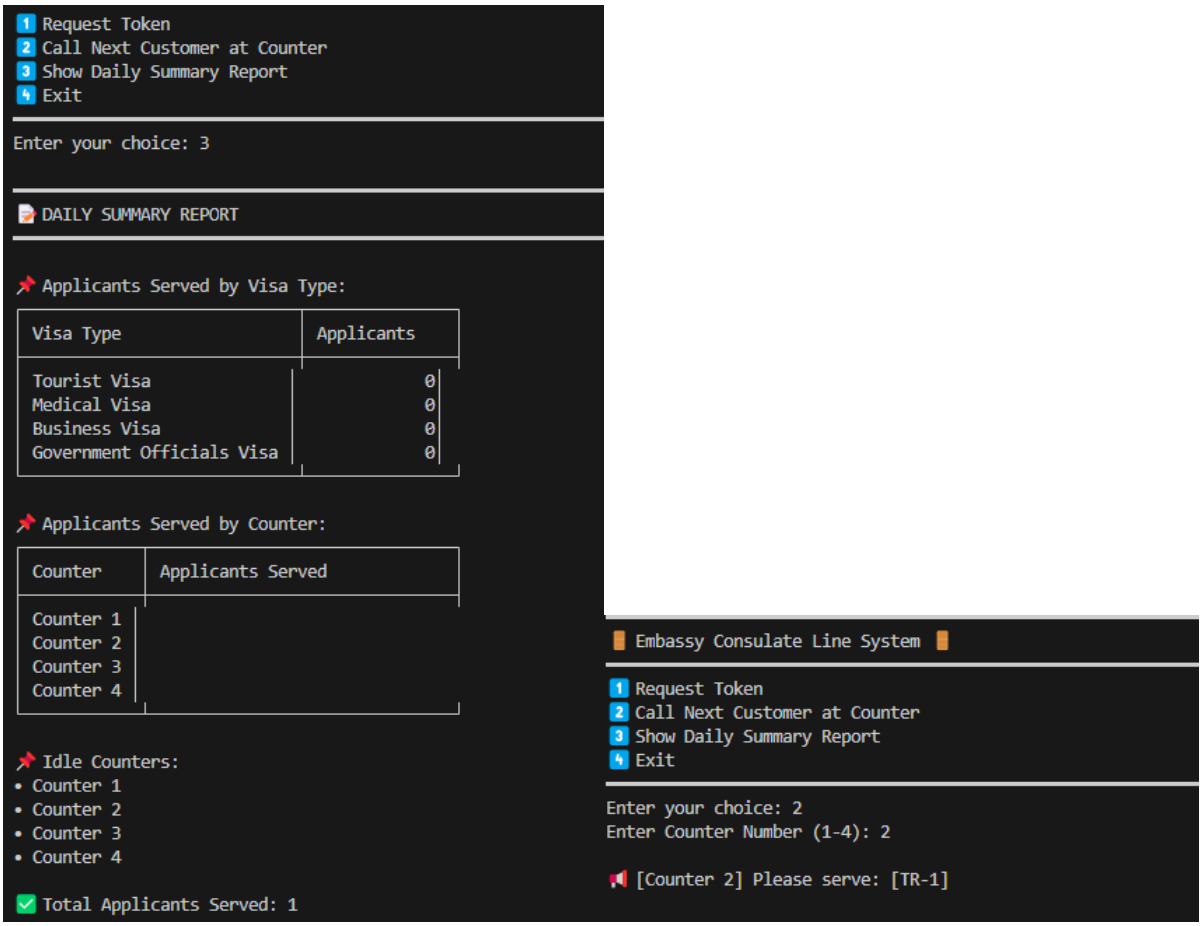
The system supports:

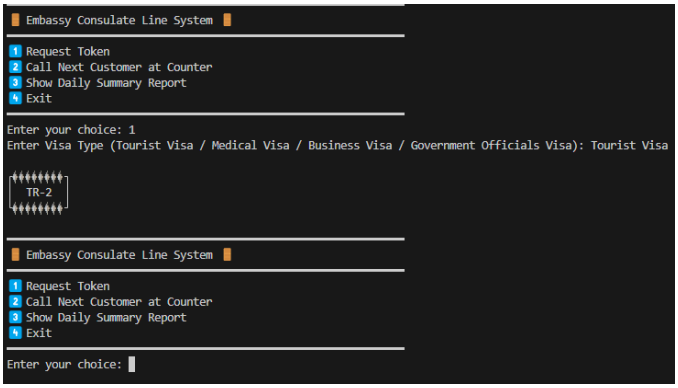
- Allowing applicants to request tokens based on their visa type.
- Managing separate queues for each visa type.
- Automatically calling the next applicant at each counter.
- Displaying a daily summary report showing how many applicants were served per visa type and per counter.
- Setting a maximum limit on applicants overall and per visa type to maintain organized service.

In simple terms, it simulates a real-world token management system commonly seen in embassies.

# #Design:

In order to improve the readability and user experience of the output, especially in the **Daily Summary Report**, I utilized a table design with proper formatting. This design ensures that the output is structured, clear, and easy to follow, which is especially important when dealing with multiple categories and counters. Below is an explanation of the key design choices:



## 2. Data Structures Used

**a) Structures:**

- We created a `struct Applicant` to store the visa type and token number together.
- Grouping these related properties made the code more readable and manageable.

**b) Arrays:**

- Arrays were used to maintain the queues for each visa type (e.g., `touristQueue[]`, `medicalQueue[]`).
- Arrays also recorded the applicants each counter served.

**c) Counters:**

- Integer counters tracked the number of applicants waiting in each visa queue.
- Additional counters recorded how many applicants were served at each counter.

**Why these structures?**

- **Arrays** were ideal because the total number of applicants and per-visa limits were fixed, making dynamic structures unnecessary.
- **Structs** helped group related information in a clean and organized way.
- Given the simplicity of the requirements, using more complex structures like linked lists wasn't needed.

## 3. Challenges Faced

- **Shifting elements in arrays:** When an applicant was served, I had to shift the entire queue left to maintain the order, which required careful handling.

- **Avoiding magic numbers:** Defining constants like `MAX_APPLICANTS`, `MAX_PER_VISA`, and `COUNTERS` early helped make the code cleaner and easier to update.
- **Selecting from the longest queue:** When a counter's primary queue was empty, finding the longest active queue required writing an efficient checking mechanism.
- **Formatting the summary report:** It took time to create a clear and neat summary table using `setw()` for proper alignment and professional formatting.

## 4.Diagram