Efficient Checkout Processes

Group members

Venon mittchel

abel langa

shakir maute

Riaan langa

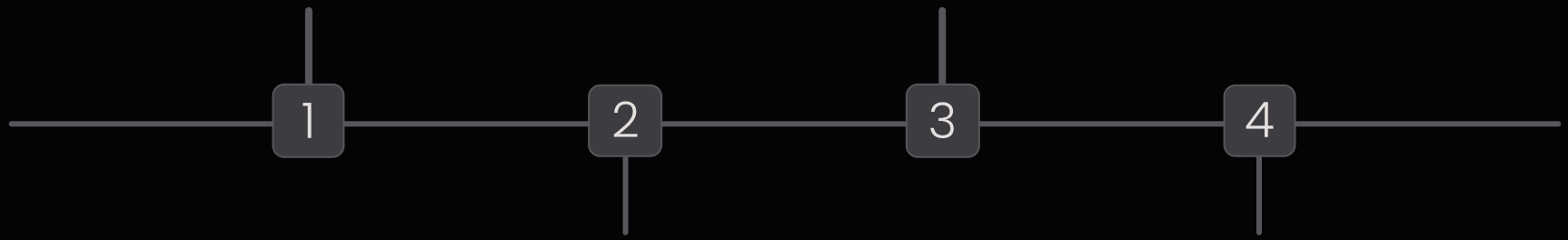
Scenario 1: One Cashier, Multiple Self-Checkout Lanes

Lane 1

The cashier oversees the processing of customers in this self-checkout lane, ensuring a smooth and efficient experience.

Lane 3

The cashier monitors this self-checkout lane, providing assistance when needed and maintaining a steady flow of customers.



Lane 2

Customers in this lane are also able to complete their transactions quickly and conveniently under the cashier's supervision.

Lane 4

This self-checkout lane also benefits from the cashier's oversight, allowing customers to complete their purchases efficiently.

Scenario 2: Each Cashier, One Checkout Lane

Lane 1

A dedicated cashier processes customers in this checkout lane, ensuring a personalized and efficient experience.

Lane 2

Another cashier is responsible for this checkout lane, providing individualized attention to each customer.

Lane 3

A third cashier handles this checkout lane, allowing for a more focused and streamlined transaction process.

Lane 4

The fourth cashier is dedicated to this checkout lane, ensuring a consistent and personalized service for all customers.

Parallel Processing of Customers

1 2 3

Scenario 1

In this scenario, a single cashier oversees the parallel processing of customers across multiple self-checkout lanes.

Scenario 2

In the second scenario, each checkout lane is handled by a dedicated cashier, allowing for the parallel processing of customers.

Efficiency Comparison

The two scenarios demonstrate different approaches to achieving efficient customer processing through parallel processing techniques.

Running the Program

Open Project

Open a new C# console application project in your IDE (e.g., Visual Studio) to run the program.

Copy Code

Copy and paste the provided code into the Program.cs file of the new project.

Run Program

Run the program to see the output demonstrating the parallel processing of customers in each scenario.

Explore Parallelism

For more advanced parallel processing, consider using the Task Parallel Library (TPL) in .NET.

Key Takeaways

Scenario 1	One cashier oversees multiple self-checkout lanes, enabling parallel processing of customers.
Scenario 2	Each checkout lane has a dedicated cashier, allowing for parallel processing of customers.
Threads	Threads are used to simulate the parallel processing of customers in both scenarios.
Parallelism	The use of threads enables the parallel processing of customers, improving efficiency and performance.
Optimization	The two scenarios demonstrate different approaches to optimizing the checkout process.