Design Document

CSCI 301 – Project 6: Simulation of a Waiting Line
Author: Dinesh Seveti
Instructor: Prof. Jie Meichsner
Due Date: November 4, 2025

Introduction

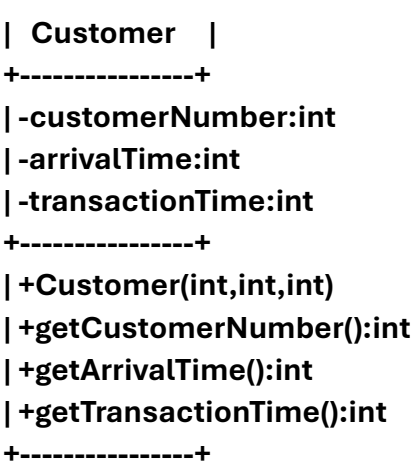
This program simulates a single waiting line of customers being served by one agent.
Customers arrive at random times and require different random service durations.
The simulation runs for a fixed time period and reports how many customers were served, how long they waited on average, and how many remained in line.
The project demonstrates use of the queue data structure, random number generation, and object-oriented design in C++.

Data Structures

Two classes are used: Customer and WaitLine.

Class	Purpose	Key Data Members
Customer	Stores one customer's information	customerNumber, arrivalTime, transactionTime
WaitLine	Manages queue and simulation	std::queue<Customer> line, numberOfArrivals, numberServed, totalTimeWaited, currentTime

UML Diagram



+----------------+
| WaitLine |
+----------------+
| -line:queue<Customer>
| -numberOfArrivals:int
| -numberServed:int
| -totalTimeWaited:int
| -currentTime:int
+----------------+
| +WaitLine()
| +simulate(int,double,int):void
+----------------+

Functions and Pseudocode

Customer(int number, int arrive, int transact)

Creates a customer object with ID number, arrival time arrive, and transaction time transact.

WaitLine::simulate(int duration, double arrivalProbability, int maxTransactionTime)

Pseudocode

```
initialize statistics to zero
transactionTimeLeft = 0
for each minute 0..duration-1:
  display current time
  if random() ≤ arrivalProbability:
     numberOfArrivals++
     transactionTime = random(1..maxTransactionTime)
     enqueue new Customer(numberOfArrivals, currentTime, transactionTime)
     print arrival message
  if transactionTimeLeft > 0:
     decrement transactionTimeLeft
     print "Customer continues"
  else if queue not empty:
     next = front of queue
     dequeue
     numberServed++
     wait = currentTime − next.arrivalTime
     totalTimeWaited += wait
     transactionTimeLeft = next.transactionTime
     print "Customer begins service..."
  else:
     print "No customers"
after loop:
  print total served, average wait, customers left
```

Structure Chart

main()

└── WaitLine::simulate(duration, arrivalProbability, maxTransactionTime)

  ├── random arrival check

  ├── queue operations

  └── statistics report