

Link Github : <https://github.com/Naufallm/Assignment-4-Pemodelan-Simulasi-Operasional-Bank.git>

## Laporan Simulasi Operasional Bank

### Pendahuluan

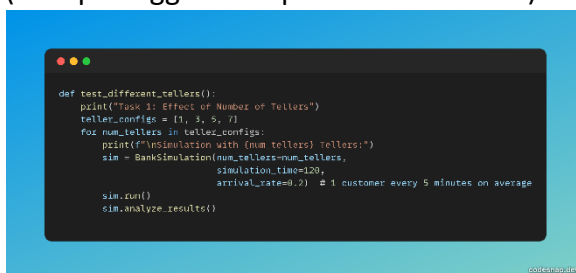
Simulasi ini bertujuan untuk menganalisis kinerja operasional bank dengan mempertimbangkan berbagai variabel seperti jumlah teller, tingkat kedatangan pelanggan, dan penerapan antrian prioritas. Simulasi dilakukan menggunakan pustaka SimPy pada Python, dengan parameter waktu simulasi selama 120 menit dan distribusi waktu kedatangan pelanggan yang mengikuti distribusi eksponensial. Waktu pelayanan pelanggan diatur secara acak antara 2 hingga 10 menit.

### Tujuan

1. Menganalisa pengaruh jumlah teller terhadap waktu tunggu layanan
2. Menganalisa performa bank selama jam sibuk dengan variasi waktu kedatangan pelanggan
3. Memvisualisasikan Tingkat utilitas teller
4. Menganalisa penerapan antrian prioritas untuk pelanggan regular dan VIP

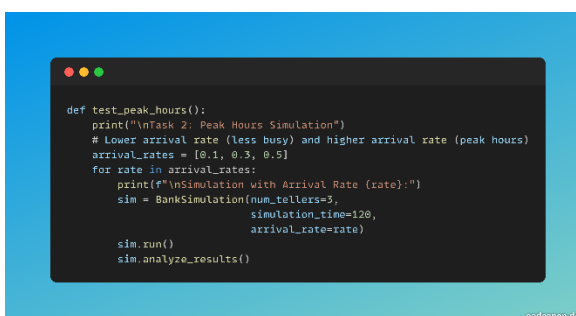
### Metodologi

1. Simulasi dengan jumlah teller bervariasi (1, 3, 5, 7) dan tingkat kedatangan tetap 0.2 (satu pelanggan setiap 5 menit rata-rata).



```
def test_different_tellers():
    print("\nTask 1: Effect of Number of Tellers")
    teller_configs = [1, 3, 5, 7]
    for num_tellers in teller_configs:
        print(f"\nSimulation with {num_tellers} tellers:")
        sim = BankSimulation(num_tellers=num_tellers,
                             simulation_time=120,
                             arrival_rate=0.2) # 1 customer every 5 minutes on average
        sim.run()
        sim.analyze_results()
```

2. Simulasi jam sibuk dengan tingkat kedatangan bervariasi (0.1, 0.3, 0.5) dan jumlah teller tetap 3



```
def test_peak_hours():
    print("\nTask 2: Peak Hours Simulation")
    # Lower arrival rate (less busy) and higher arrival rate (peak hours)
    arrival_rates = [0.1, 0.3, 0.5]
    for rate in arrival_rates:
        print(f"\nSimulation with Arrival Rate {rate}:")
        sim = BankSimulation(num_tellers=3,
                             simulation_time=120,
                             arrival_rate=rate)
        sim.run()
        sim.analyze_results()
```

3. Simulasi antrian prioritas dengan 3 teller, tingkat kedatangan 0.3, dan 20% pelanggan diklasifikasikan sebagai VIP

```
class PriorityBankSimulation:
    def __init__(self, num_tellers, simulation_time, arrival_rate):
        self.env = simpy.Environment()
        self.num_tellers = num_tellers
        self.simulation_time = simulation_time
        self.arrival_rate = arrival_rate
        self.teller_resource = simpy.PriorityResource(self.env, capacity=num_tellers)
        self.waiting_times = {
            'regular': [],
            'vip': []
        }
        self.service_times = []
        self.agent_utilization = []

    def customer(self, name, service_time, is_vip=False):
        # Record arrival time
        arrival_time = self.env.now

        # Determine priority (lower number = higher priority)
        priority = 0 if is_vip else 1

        # Request a teller with priority
        with self.teller_resource.request(priority=priority) as request:
            yield request

        # Wait time calculation
        wait_time = self.env.now - arrival_time

        # Store wait time based on customer type
        if is_vip:
            self.waiting_times['vip'].append(wait_time)
        else:
            self.waiting_times['regular'].append(wait_time)

        # Service time
        yield self.env.timeout(service_time)

        # Record service time
        self.service_times.append(service_time)

    def customer_generator(self):
        customer_id = 0
        while self.env.now < self.simulation_time:
            # Exponential distribution for inter-arrival times
            yield self.env.timeout(random.expovariate(self.arrival_rate))

            # Random service time (between 2-10 minutes)
            service_time = random.uniform(2, 10)

            customer_id += 1

            # 20% of customers are VIP
            is_vip = random.random() < 0.2

            # Process the customer
            customer_type = 'VIP' if is_vip else 'Regular'
            self.env.process(self.customer(f'{customer_type} Customer (customer id)',
                                         service_time,
                                         is_vip=is_vip))

    def record_agent_utilization(self):
        while self.env.now < self.simulation_time:
            # Calculate current utilization
            utilization = len(self.teller_resource.users) / self.num_tellers
            self.agent_utilization.append((self.env.now, utilization))
            yield self.env.timeout(1) # Check every minute

    def run(self):
        # Start customer generator and utilization tracker
        self.env.process(self.customer_generator())
        self.env.process(self.record_agent_utilization())

        # Run the simulation
        self.env.run(until=self.simulation_time)

    def analyze_results(self):
        # Analyze waiting times for regular and VIP customers
        print("\nPriority Queue Analysis:")

        # Regular Customers
        if self.waiting_times['regular']:
            print("Regular Customers:")
            print(f"Average Waiting Time: {np.mean(self.waiting_times['regular']):.2f} minutes")
            print(f"Maximum Waiting Time: {np.max(self.waiting_times['regular']):.2f} minutes")
            print(f"Total Regular Customers: {len(self.waiting_times['regular'])}")

        # VIP Customers
        if self.waiting_times['vip']:
            print("VIP Customers:")
            print(f"Average Waiting Time: {np.mean(self.waiting_times['vip']):.2f} minutes")
            print(f"Maximum Waiting Time: {np.max(self.waiting_times['vip']):.2f} minutes")
            print(f"Total VIP Customers: {len(self.waiting_times['vip'])}")

        # Visualize agent utilization
        if self.agent_utilization:
            times, utilizations = zip(*self.agent_utilization)
            plt.figure(figsize=(10, 5))
            plt.plot(times, utilizations)
            plt.title('Agent Utilization Over Time')
            plt.xlabel('Time (minutes)')
            plt.ylabel('Utilization Rate')
            plt.ylim(0, 1)
            plt.show()
```

## Hasil Analisa

- 1 **Teller:** Rata-rata waktu tunggu 8.12 menit, maksimum 15.08 menit.
- 2 **Jam Sibuk:** Waktu tunggu meningkat seiring tingkat kedatangan.
- 3 **Prioritas:** VIP menunggu lebih singkat, reguler lebih lama.
- 4 **Utilisasi:** Tinggi pada teller terbatas, fluktuatif pada simulasi.

**Kesimpulan**

Lebih banyak teller dan penyesuaian pada jam sibuk mengurangi waktu tunggu.  
Antrian prioritas efektif untuk VIP, tapi memengaruhi pelanggan reguler.