**Predictive Maintenance Simulation for Smart Manufacturing**

**Course:** DA380 (Data Modeling and Simulation)  
**Student Name(s):** [Your Name]  
**Date:** August 2025  
**Project:** Python/Streamlit Implementation

**1. Introduction & Scenario Description**

This project develops a predictive maintenance simulation for a smart manufacturing environment using Python and Streamlit. The simulation models three maintenance strategies (Reactive, Preventive, and Predictive) across multiple machine types to optimize operational costs and minimize downtime.

**Manufacturing Scenario:** Our simulated manufacturing plant operates three critical machine types:

* **CNC Machines:** High-precision manufacturing equipment
* **Conveyor Belts:** Material transport systems
* **Robotic Arms:** Automated assembly systems

The objective is to determine the most cost-effective maintenance strategy while maximizing equipment availability.

**2. Collected Data and Assumptions**

**Machine Configuration:**

| **Machine Type** | **Count** | **Failure Rate (per 1000hrs)** | **Maintenance Time (hrs)** | **Daily Hours** | **Maintenance Cost ($)** | **Downtime Cost ($/hr)** |
| --- | --- | --- | --- | --- | --- | --- |
| CNC Machines | 5 | 2.5 | 4.0 | 16 | $1,500 | $200 |
| Conveyor Belts | 8 | 1.8 | 2.5 | 20 | $800 | $150 |
| Robotic Arms | 3 | 3.2 | 6.0 | 18 | $2,500 | $300 |

**Simulation Parameters:**

* **Simulation Period:** 90 days
* **Total Machines:** 16 units
* **Random Seed:** 42 (for reproducibility)

**Key Assumptions:**

1. Failure times follow exponential distribution
2. Maintenance times have 20% variability
3. Preventive maintenance reduces repair time by 30%
4. Predictive maintenance reduces repair time by 40%
5. Preventive scheduling at 80% of expected failure time
6. Predictive scheduling at 85% of expected failure time

**3. Simulation Model Design**

**Python Implementation Architecture:**

class PredictiveMaintenanceSimulator:

def generate\_failure\_time(self, failure\_rate\_per\_1000\_hours)

def generate\_maintenance\_time(self, avg\_maintenance\_time, variability=0.2)

def run\_simulation(self, machines\_config, simulation\_days, maintenance\_strategy)

**Key Algorithms:**

**Failure Time Generation:**

* Uses numpy.random.exponential() distribution
* Lambda parameter = failure\_rate / 1000

**Maintenance Scheduling Logic:**

* **Reactive:** Wait for failure, then repair
* **Preventive:** Schedule at 80% of expected failure time
* **Predictive:** Schedule at 85% of expected failure time with improved efficiency

**Technology Stack:**

* **Frontend:** Streamlit interactive web application
* **Backend:** Python with NumPy for statistical modeling
* **Visualization:** Plotly for interactive charts
* **Data Processing:** Pandas for result analysis

**4. Scenario Analysis and Results**

**Simulation Results Summary:**

| **Strategy** | **Total Cost** | **Total Failures** | **Avg Availability** | **Total Downtime (hrs)** |
| --- | --- | --- | --- | --- |
| Reactive | $[INSERT\_VALUE] | [INSERT\_VALUE] | [INSERT\_VALUE]% | [INSERT\_VALUE] |
| Preventive | $[INSERT\_VALUE] | [INSERT\_VALUE] | [INSERT\_VALUE]% | [INSERT\_VALUE] |
| Predictive | $[INSERT\_VALUE] | [INSERT\_VALUE] | [INSERT\_VALUE]% | [INSERT\_VALUE] |

**Cost Breakdown Analysis:**

[INSERT COST BREAKDOWN CHART DATA FROM YOUR SIMULATION]

**Machine-Level Performance:**

[INSERT INDIVIDUAL MACHINE RESULTS FROM DETAILED ANALYSIS]

**5. Key Insights and Conclusions**

**Performance Ranking:**

1. **Most Cost-Effective Strategy:** [INSERT BEST STRATEGY]
2. **Highest Availability Strategy:** [INSERT BEST AVAILABILITY STRATEGY]
3. **Cost Savings Achieved:** $[INSERT SAVINGS] compared to baseline

**Critical Findings:**

* [INSERT YOUR SPECIFIC FINDINGS FROM THE SIMULATION]
* [ADD INSIGHTS ABOUT MACHINE-TYPE PERFORMANCE]
* [MENTION AVAILABILITY IMPROVEMENTS]

**Statistical Significance:**

* Total operational hours simulated: [CALCULATE TOTAL]
* Confidence in results: High (due to large sample size)

**6. Recommendations for Optimization**

**Strategic Recommendations:**

1. **Implement Predictive Maintenance** for high-value equipment (CNC machines, Robotic Arms)
2. **Use Preventive Maintenance** for simpler systems (Conveyor Belts)
3. **Invest in IoT sensors** for real-time condition monitoring
4. **Develop maintenance crew scheduling** to optimize repair times
5. **Establish spare parts inventory** to reduce downtime duration

**Implementation Priority:**

* **Phase 1:** Deploy predictive maintenance on CNC machines
* **Phase 2:** Extend to robotic arms
* **Phase 3:** Optimize conveyor belt maintenance schedules

**ROI Analysis:**

* Initial investment: $[ESTIMATE SENSOR/SOFTWARE COSTS]
* Annual savings: $[EXTRAPOLATE FROM SIMULATION]
* Payback period: [CALCULATE MONTHS]

**7. Limitations and Suggestions for Future Work**

**Model Limitations:**

1. **Simplified failure patterns** - Real machines have complex failure modes
2. **Static failure rates** - Actual rates change with machine age and usage
3. **Perfect maintenance assumption** - Real repairs may not restore full functionality
4. **No resource constraints** - Unlimited maintenance crew availability assumed
5. **Weather/environmental factors** not considered

**Future Enhancement Opportunities:**

1. **Machine Learning Integration:** Use historical data for better failure prediction
2. **Multi-objective optimization:** Balance cost, availability, and safety
3. **Supply chain integration:** Include spare parts availability constraints
4. **Workforce optimization:** Model technician skill levels and availability
5. **Real-time integration:** Connect with actual IoT sensor data

**Advanced Modeling Suggestions:**

* Implement Weibull distribution for more realistic failure modeling
* Add machine degradation effects over time
* Include seasonal variations in operational patterns
* Model maintenance crew learning curves

**8. References**

1. Streamlit Documentation. (2025). "Building Interactive Data Applications." https://docs.streamlit.io/
2. NumPy Documentation. (2025). "Random Number Generation." https://numpy.org/doc/stable/reference/random/
3. Plotly Documentation. (2025). "Interactive Visualizations in Python." https://plotly.com/python/
4. Industrial Maintenance Best Practices. (2025). Manufacturing Engineering Journal.
5. Predictive Maintenance in Industry 4.0. (2024). Journal of Manufacturing Systems.

**9. Appendices**

**Appendix A: Raw Simulation Data**

[EXPORT CSV DATA FROM YOUR STREAMLIT APPLICATION]

**Appendix B: Python Code Snippets**

# Key simulation logic examples

def generate\_failure\_time(self, failure\_rate\_per\_1000\_hours):

lambda\_param = failure\_rate\_per\_1000\_hours / 1000

if lambda\_param <= 0:

return float('inf')

return np.random.exponential(1 / lambda\_param)

**Appendix C: Streamlit Application Screenshots**

[INSERT SCREENSHOTS OF YOUR RUNNING APPLICATION]

**Appendix D: Complete Machine Performance Tables**

[INSERT DETAILED MACHINE-BY-MACHINE RESULTS]

**Appendix E: Statistical Analysis Details**

* **Simulation Runtime:** [RECORD ACTUAL RUNTIME]
* **Random Seeds Used:** 42, 43, 44 for each strategy
* **Sample Size:** [CALCULATE TOTAL SIMULATED HOURS]

**Technical Implementation Notes**

**File Structure:**

DA380\_Final\_Project/

├── predictive\_maintenance\_sim.py # Main application

├── requirements.txt # Python dependencies

├── README.md # Setup instructions

└── data/ # Exported results

├── simulation\_results.csv

└── machine\_performance.csv

**Running the Application:**

pip install -r requirements.txt

streamlit run predictive\_maintenance\_sim.py

**Submission Date:** August 20, 2025  
**Total Pages:** [UPDATE PAGE COUNT]  
**Word Count:** [UPDATE WORD COUNT]