

**Newspaper Seller Case Study**

## **Problem Definition:**

The newspaper seller wants to determine how many newspapers he should buy each day in order to **maximize his profits**. His present method of determining the quantity of Newspaper is based upon his best guess or estimate of the daily demand for the day news.

## **Assumptions and Parameters:**

* **Selling Price (P):** $0.50 per newspaper
* **Cost Price (C):** $0.33 per newspaper
* **Scrap Value (S):** $0.05 per unsold newspaper
* **X:** Quantity of newspapers purchased
* **d:** Daily demand
* **Z:** Daily profit

## **Day Type Probability Distribution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Type of Newsday | Probability | Cumulative | Random Number Range |
| Good | 0.35 | 0.35 | 0 ≤ R < 0.35 |
| Fair | 0.45 | 0.80 | 0.35 ≤ R < 0.80 |
| Poor | 0.20 | 1.00 | 0.80 ≤ R ≤ 1 |

## **Demand Distribution:**

(For demand values: 40, 50, 60, 70, 80, 90, 100 newspapers)

You will need to calculate the Cumulative Distribution Function (CDF) for demand for each type of day (Good, Fair, Poor) using your Excel data. You can present each distribution in a table format like:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Demand Prob. Dist. | | |
| Demand | Good | Fair | Poor |
| 40 | 0.03 | 0.10 | 0.44 |
| 50 | 0.05 | 0.18 | 0.22 |
| 60 | 0.15 | 0.40 | 0.16 |
| 70 | 0.20 | 0.20 | 0.12 |
| 80 | 0.35 | 0.08 | 0.06 |
| 90 | 0.15 | 0.04 | 0.00 |
| 100 | 0.07 | 0.00 | 0.00 |

## **Model Formulation:**

***Key Equations:***

1. Revenue Sales = min(d, X) × P
2. Excess Demand = max(X, d) − d
3. Profit per newspaper sold = P − C
4. Lost Profit = (X < d) ? (d − X) × (P − C)
5. Number of Scrap = max(X − d, 0)
6. Scrap Salvage = Number of Scrap × S
7. Cost of Newspapers = X × C
8. Daily Profit (Z) = Revenue Sales − Cost of Newspapers − Lost Profit + Scrap Salvage

## **Objective**

Use simulation (e.g., random number generation for day type and demand) to estimate the optimal number of newspapers to purchase daily (X) that maximizes the average profit Z (X, d).

## **Implementation**

#### **Excel Sheet:**

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

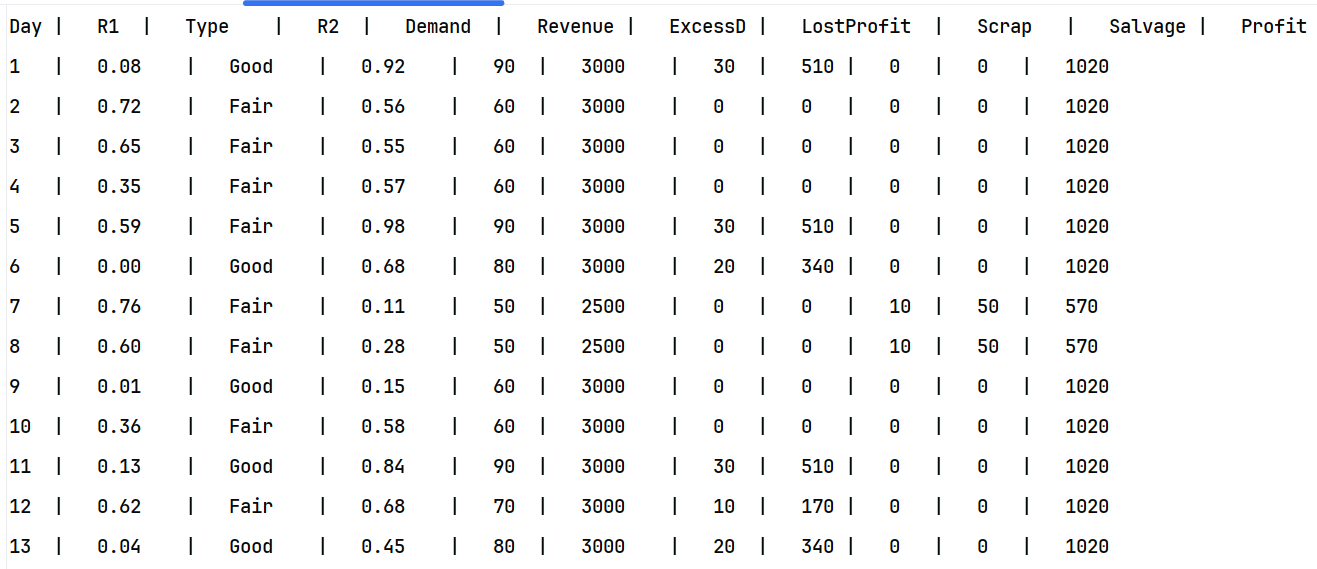
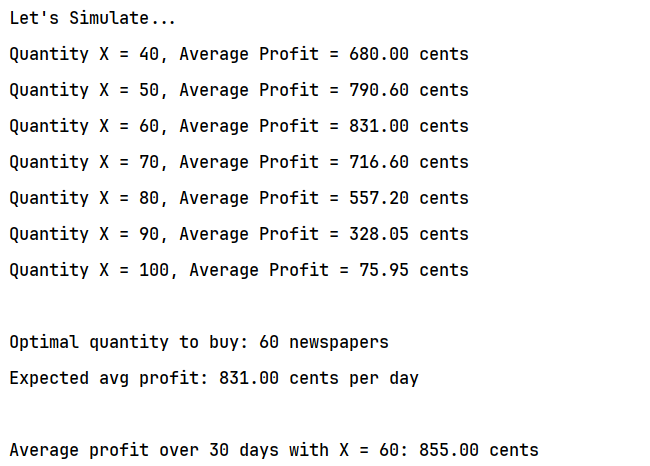
#### **C++ Code:**

#include<bits/stdc++.h>  
using namespace std;  
  
constexpr int days = 30;  
constexpr int quantity = 70;  
constexpr int paper\_cost = 33;  
constexpr int iterations = 10;  
constexpr float scrap\_sale = 5;  
constexpr float paper\_sell = 50;  
  
*/// Types of News day*constexpr float GOOD = 0.35;  
constexpr float FAIR = 0.45;  
constexpr float POOR = 0.20;  
  
float cumulative\_prop[3] = {  
 GOOD,  
 GOOD + FAIR,  
 GOOD + FAIR + POOR,  
};  
  
struct RD {  
 float lower, upper;  
} RandDist[3]{  
 {0.0, GOOD - 0.01},  
 {GOOD, GOOD + FAIR - 0.01},  
 {GOOD + FAIR, GOOD + FAIR + POOR}  
};  
  
struct DayData {  
 int day;  
 double R1;  
 string type;  
 double R2;  
 int demand;  
 int revenue\_sales;  
 int excess\_demand;  
 int lost\_profit\_excess;  
 int num\_scrap;  
 int salvage\_scrap;  
 int daily\_profit;  
};  
  
int getDemand(const string& dayType, const double R) {  
 if (dayType == "Good") {  
 if (R < 0.03) return 40;  
 if (R < 0.08) return 50;  
 if (R < 0.23) return 60;  
 if (R < 0.43) return 70;  
 if (R < 0.78) return 80;  
 if (R < 0.93) return 90;  
 return 100;  
 }  
  
 if (dayType == "Fair") {  
 if (R < 0.10) return 40;  
 if (R < 0.28) return 50;  
 if (R < 0.68) return 60;  
 if (R < 0.88) return 70;  
 if (R < 0.96) return 80;  
 return 90;  
 }  
  
 if (dayType == "Poor") {  
 if (R < 0.44) return 40;  
 if (R < 0.66) return 50;  
 if (R < 0.82) return 60;  
 if (R < 0.94) return 70;  
 return 80;  
 }  
 return {};  
}  
  
vector<DayData> simulate(int X, int N) {  
 vector<DayData> results;  
 random\_device rd;  
 mt19937 gen(rd());  
 uniform\_real\_distribution<> dis(0.0, 1.0);  
  
 for (int day = 1; day <= N; ++day) {  
 double R1 = dis(gen);  
 string type;  
 if (R1 < cumulative\_prop[0]) type = "Good";  
 else if (R1 < cumulative\_prop[1]) type = "Fair";  
 else type = "Poor";  
  
 double R2 = dis(gen);  
 int demand = getDemand(type, R2);  
  
 int revenue\_sales = min(X, demand) \* paper\_sell;  
 int excess\_demand = max(demand - X, 0);  
 int lost\_profit\_excess = excess\_demand \* (paper\_sell - paper\_cost);  
 int num\_scrap = max(X - demand, 0);  
 int salvage\_scrap = num\_scrap \* scrap\_sale;  
 int daily\_profit = revenue\_sales + salvage\_scrap - X \* paper\_cost;  
  
 DayData data = {day, R1, type, R2, demand, revenue\_sales,

excess\_demand, lost\_profit\_excess, num\_scrap, salvage\_scrap, daily\_profit};  
 results.push\_back(data);  
 }  
 return results;  
}  
  
void writeTableToFile(const vector< DayData > &results, const string filename) {  
 ofstream outfile(filename);  
 if (!outfile) return void (cerr << "NOOOOOOOOOOOOOOOO such File: " << filename << endl);  
  
 outfile << "Day\t|\tR1\t|\tType\t|\tR2\t|\tDemand\t|\tRevenue\t|\tExcessD\t|\tLostProfit\t|\tScrap\t|\tSalvage\t|\tProfit\n";  
 for (const auto &data: results) {  
 outfile << data.day << "\t| \t"  
 << fixed << setprecision(2) << data.R1 << "\t| \t"  
 << data.type << "\t| \t"  
 << fixed << setprecision(2) << data.R2 << "\t| \t"  
 << data.demand << "\t| \t"  
 << data.revenue\_sales << "\t| \t"  
 << data.excess\_demand << "\t| \t"  
 << data.lost\_profit\_excess << "\t| \t"  
 << data.num\_scrap << "\t| \t"  
 << data.salvage\_scrap << "\t| \t"  
 << data.daily\_profit << "\n";  
 }  
 outfile.close();  
}  
  
int main() {  
 int N\_large = 1000;  
 vector< int > X\_values = {40, 50, 60, 70, 80, 90, 100};  
 double best\_avg\_profit = -numeric\_limits< double >::*max*();  
 int best\_X = -1;  
  
 cout << "Let's Simulate...\n";  
 for (int X: X\_values) {  
 auto results = simulate(X, N\_large);  
 double total\_profit = 0;  
  
 for (const auto &data: results) total\_profit += data.daily\_profit;  
  
 double avg\_profit = total\_profit / N\_large;  
 cout << "Quantity X = " << X << ", Average Profit = " << fixed << setprecision(2)

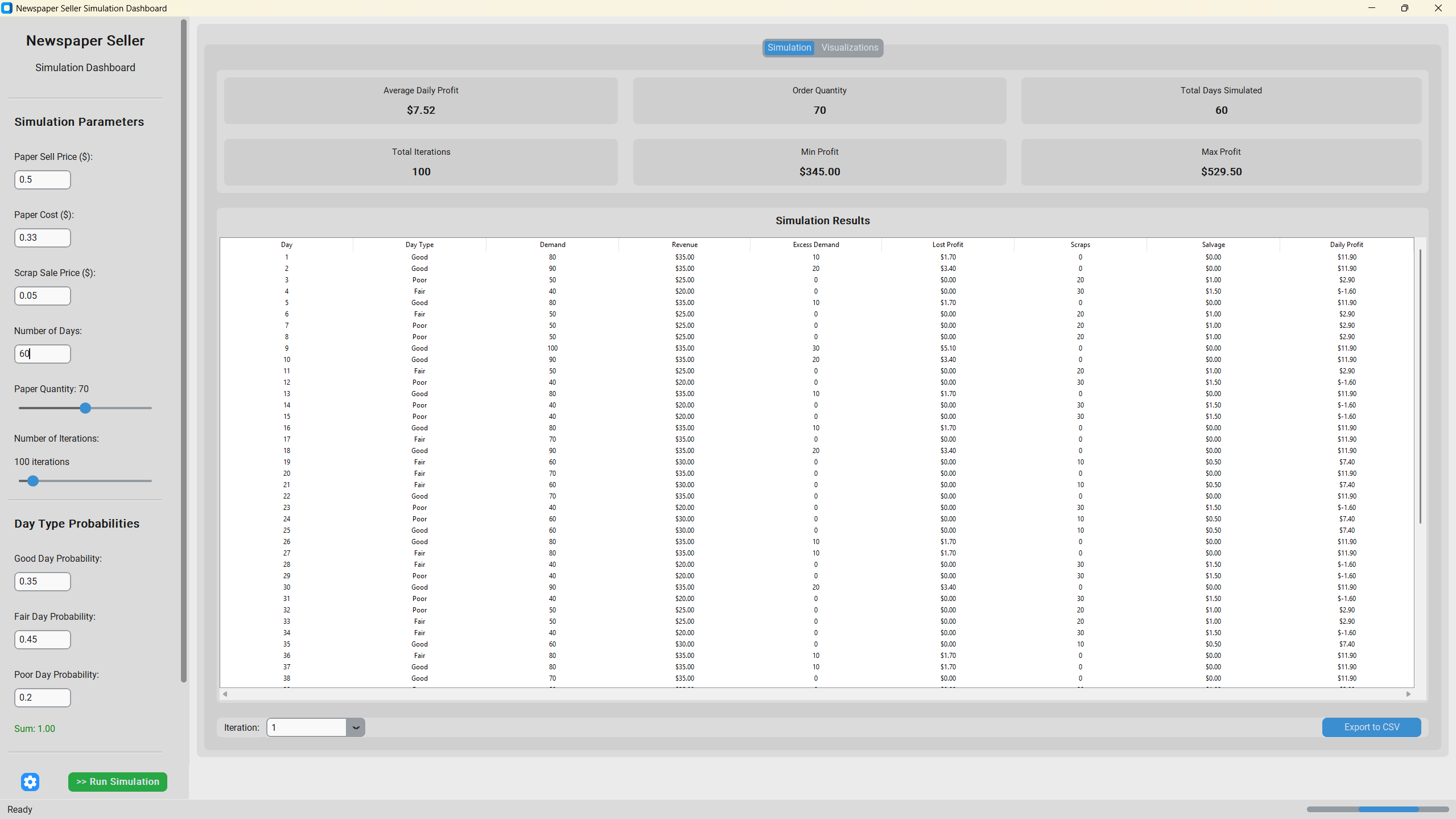
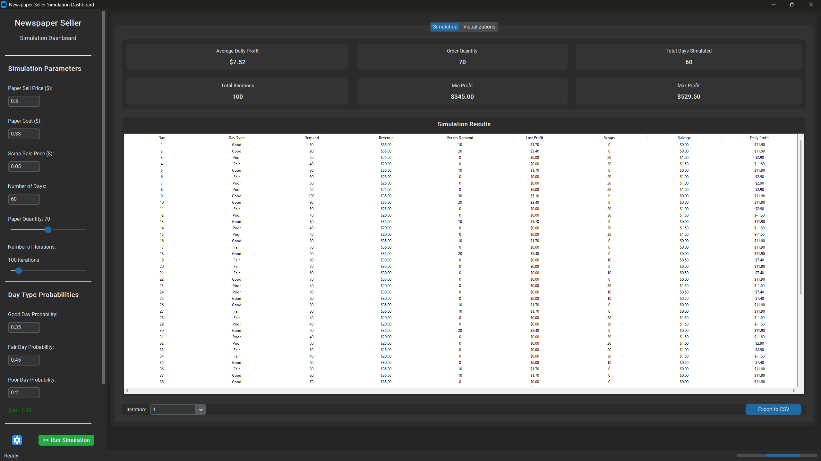
<< avg\_profit << " cents\n";  
 if (avg\_profit > best\_avg\_profit) best\_avg\_profit = avg\_profit, best\_X = X;  
 }  
  
 cout << "\nOptimal quantity to buy: " << best\_X << " newspapers\n";  
 cout << "Expected avg profit: " << fixed << setprecision(2) << best\_avg\_profit << " cents per day\n";  
  
 auto table\_results = simulate(best\_X, days);  
 writeTableToFile(table\_results, "newspaper\_simulation.txt");  
  
 double table\_total\_profit = 0;  
 for (const auto &data: table\_results) {  
 table\_total\_profit += data.daily\_profit;  
 }  
 double table\_avg\_profit = table\_total\_profit / days;  
 cout << "\nAverage profit over " << days << " days with X = " << best\_X << ": "

<< fixed << setprecision(2) <<  
 table\_avg\_profit << " cents\n";  
  
 return 0;  
}



**Run**

## **GUI Presentation**

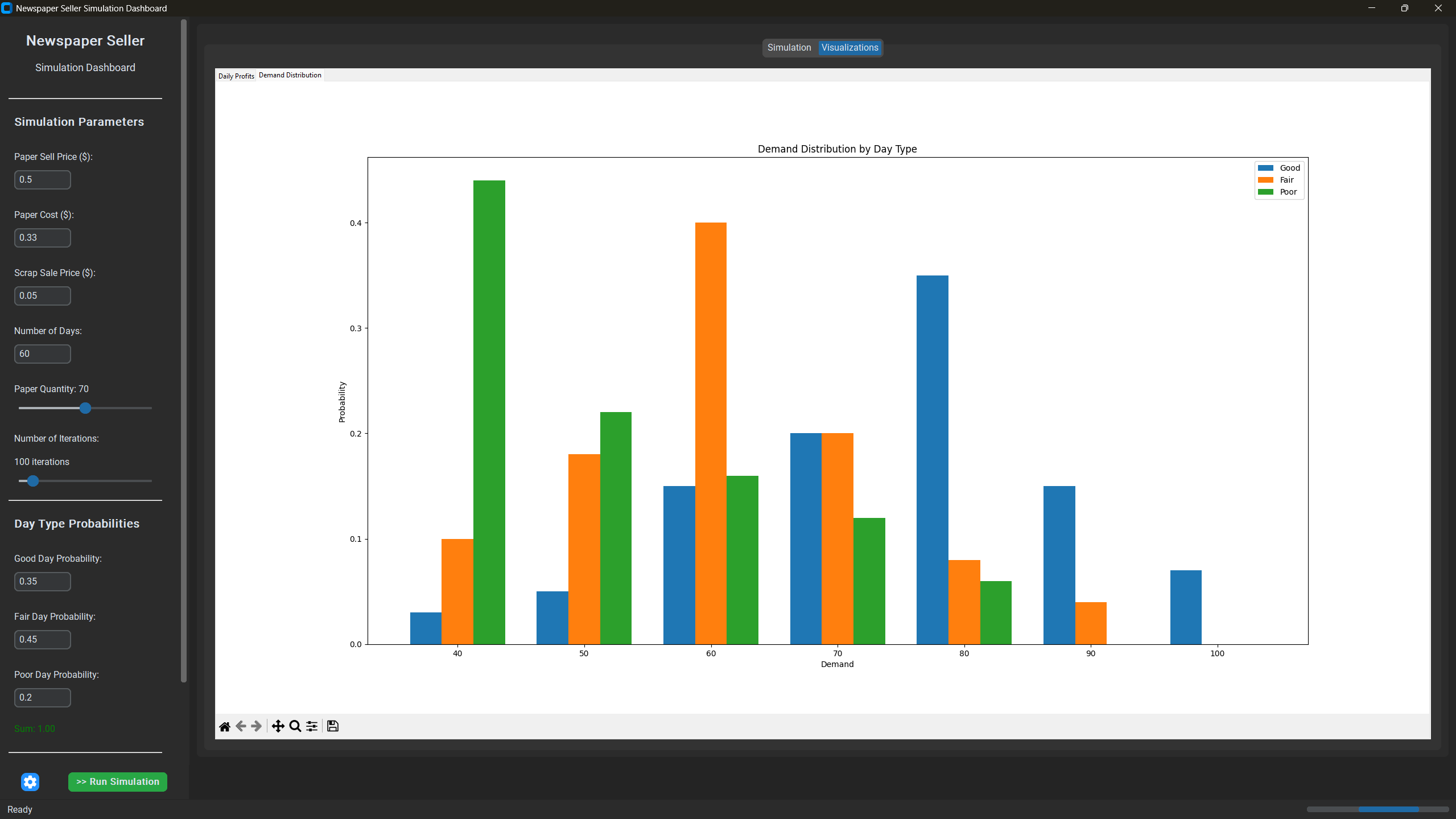
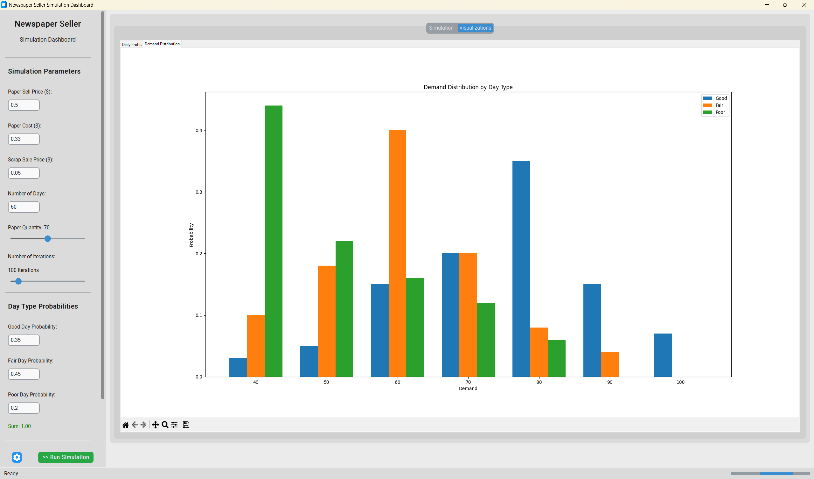


A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.



## **Flowchart**

A black and white image of a cross

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

## **Results and Conclusion**

* Present results in a table/graph: Quantity vs. Average Profit.
* Interpret how demand patterns and day types of influence profitability.