ORATHON

KNAPSACK SOLVER

PRESENTED TO
OR Club

PRESENTED BY
Team 6 (One&Zero)

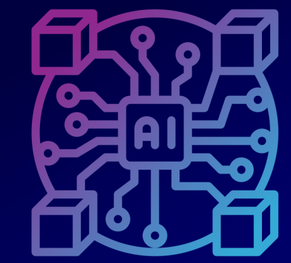
ORATHON

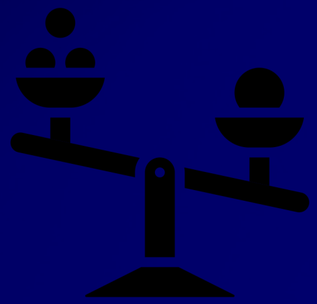
Problematic

Adam wants to import materials with his own boat, but he needs to choose items with the best value and minimal weight. He is unsure of how to determine the optimal combination of items to take given limited space on his boat, risking incurring unnecessary expenses. How can Adam solve this problem and ensure a successful import operation?

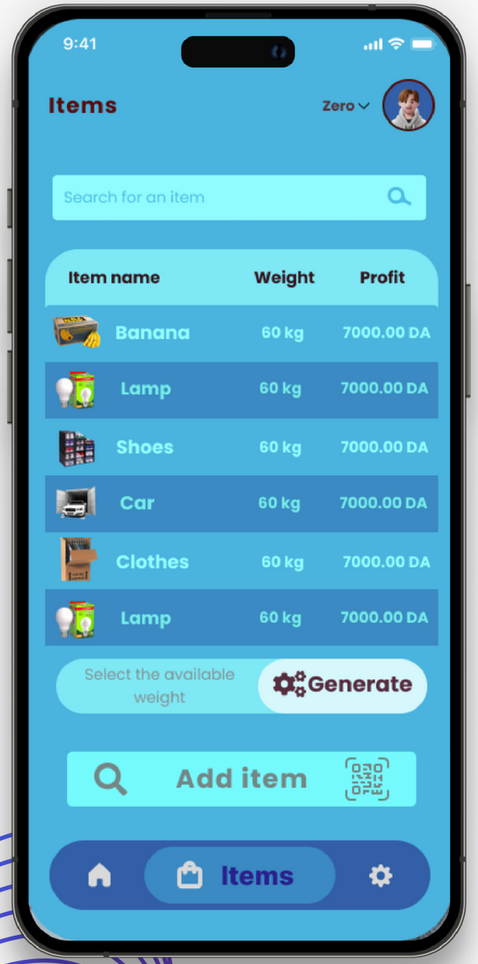
Our Solution

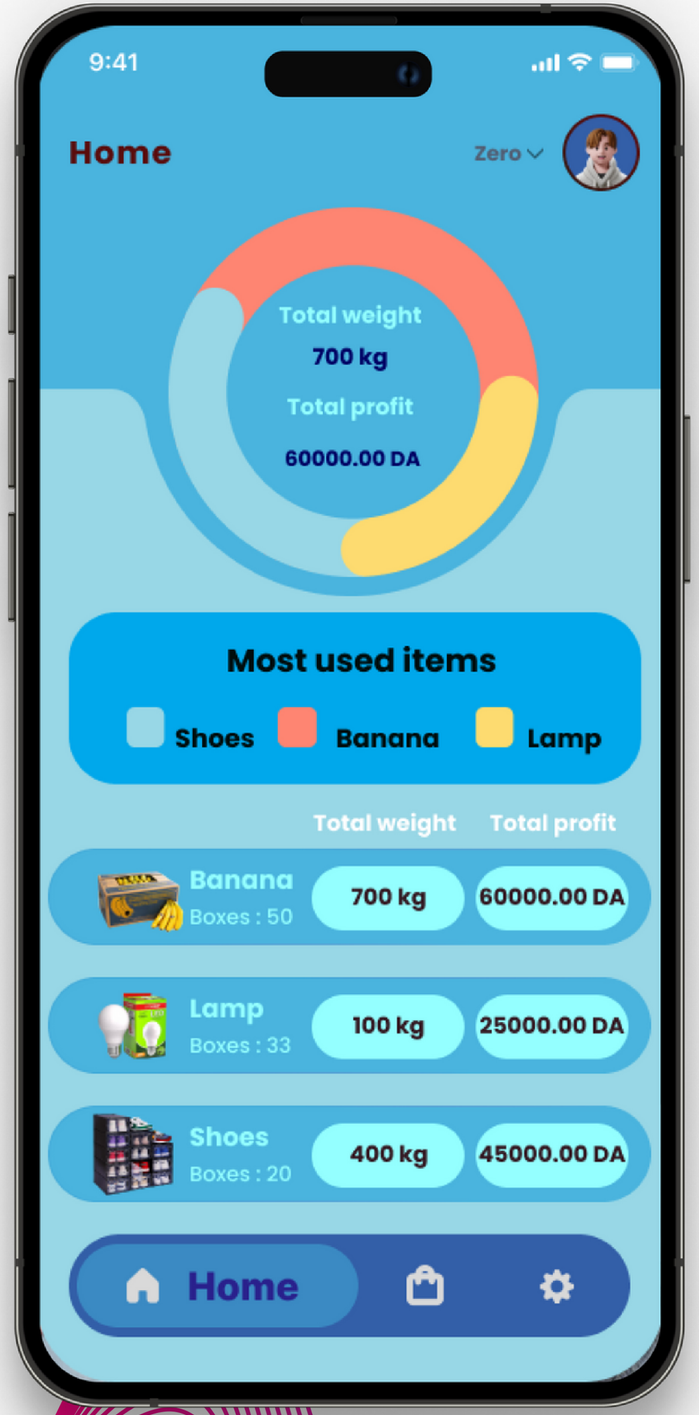
Its an app with a custom algorithm designed to help him choose the optimal combination of items. The app would consider factors such as the weight and value of each item, as well as the available space on the boat. This would help Adam maximize his profits, while also ensuring that the boat is not overloaded. By using this app, Adam can easily plan his import operation, save money, and avoid potential damage to his boat.

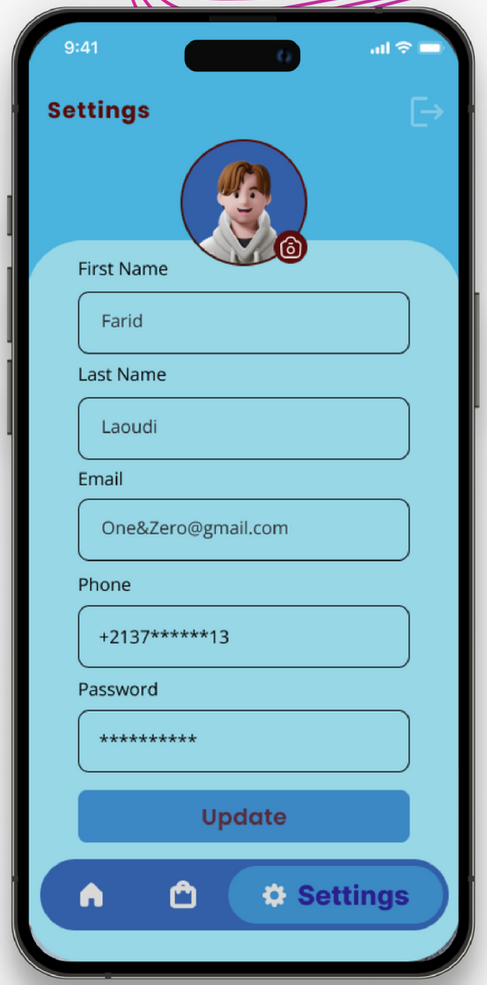


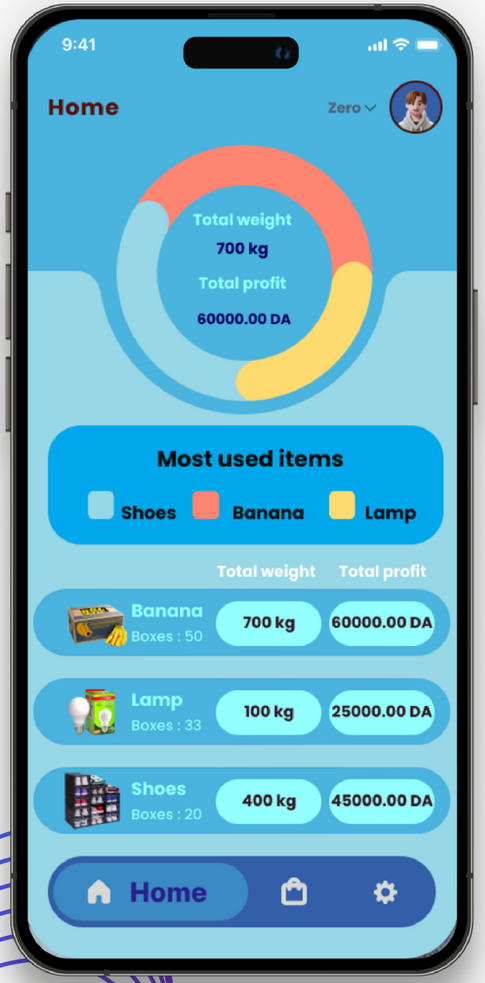


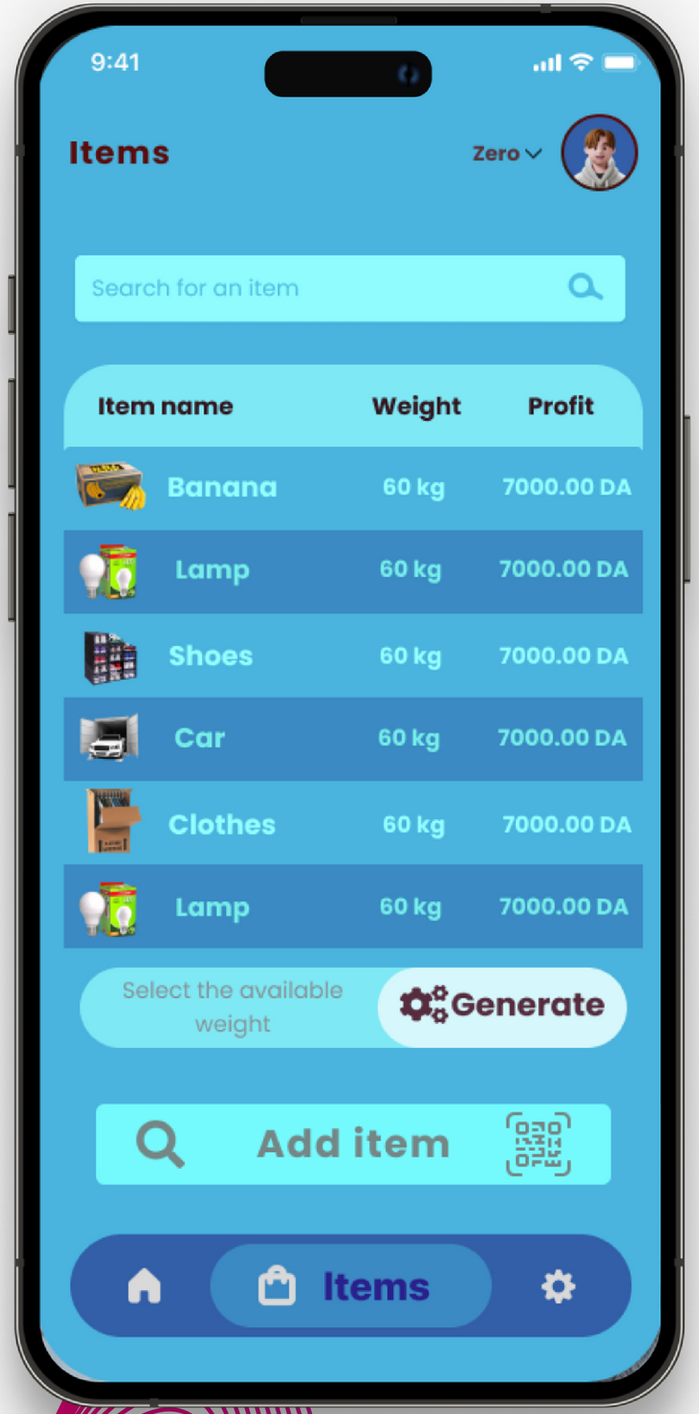
The Prototype

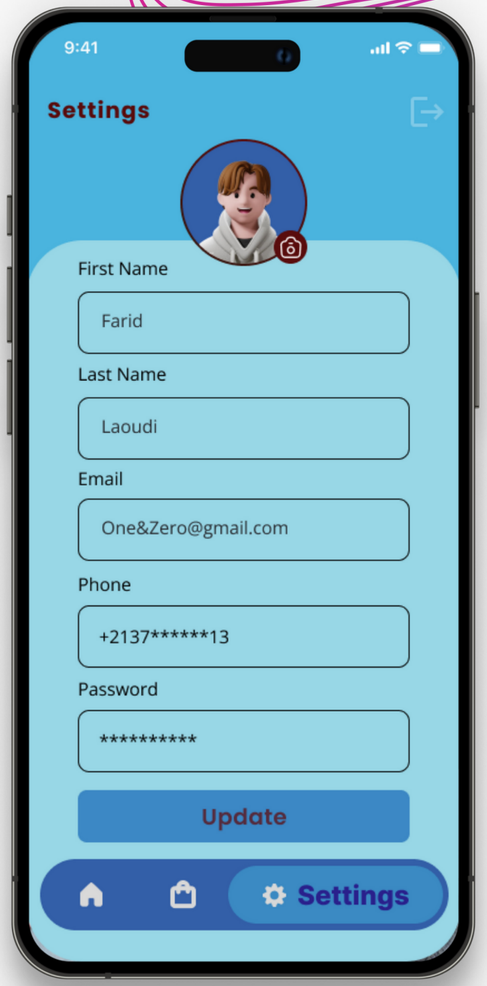

9:41
Items
Zero
Search for an item
Item name
Weight
Profit
Banana 60 kg 7000.00 DA
Lamp 60 kg 7000.00 DA
Shoes 60 kg 7000.00 DA
Car 60 kg 7000.00 DA
Clothes 60 kg 7000.00 DA
Lamp 60 kg 7000.00 DA
Select the available weight
Generate
Add item
Items

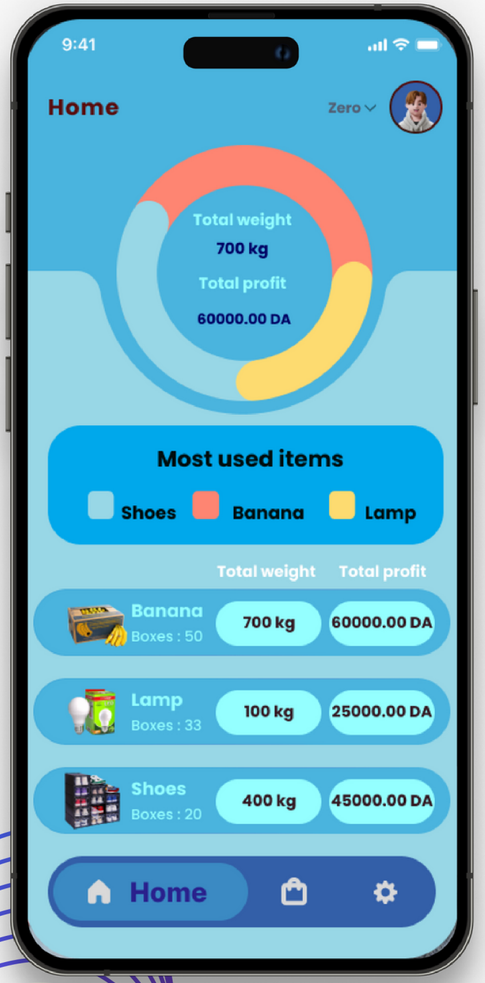

9:41
Home
Zero
Total weight
700 kg
Total profit
60000.00 DA
Most used items
Shoes
Banana
Lamp
Total weight
Total profit
Banana
Boxes : 50
700 kg
60000.00 DA
Lamp
Boxes : 33
100 kg
25000.00 DA
Shoes
Boxes : 20
400 kg
45000.00 DA
Home

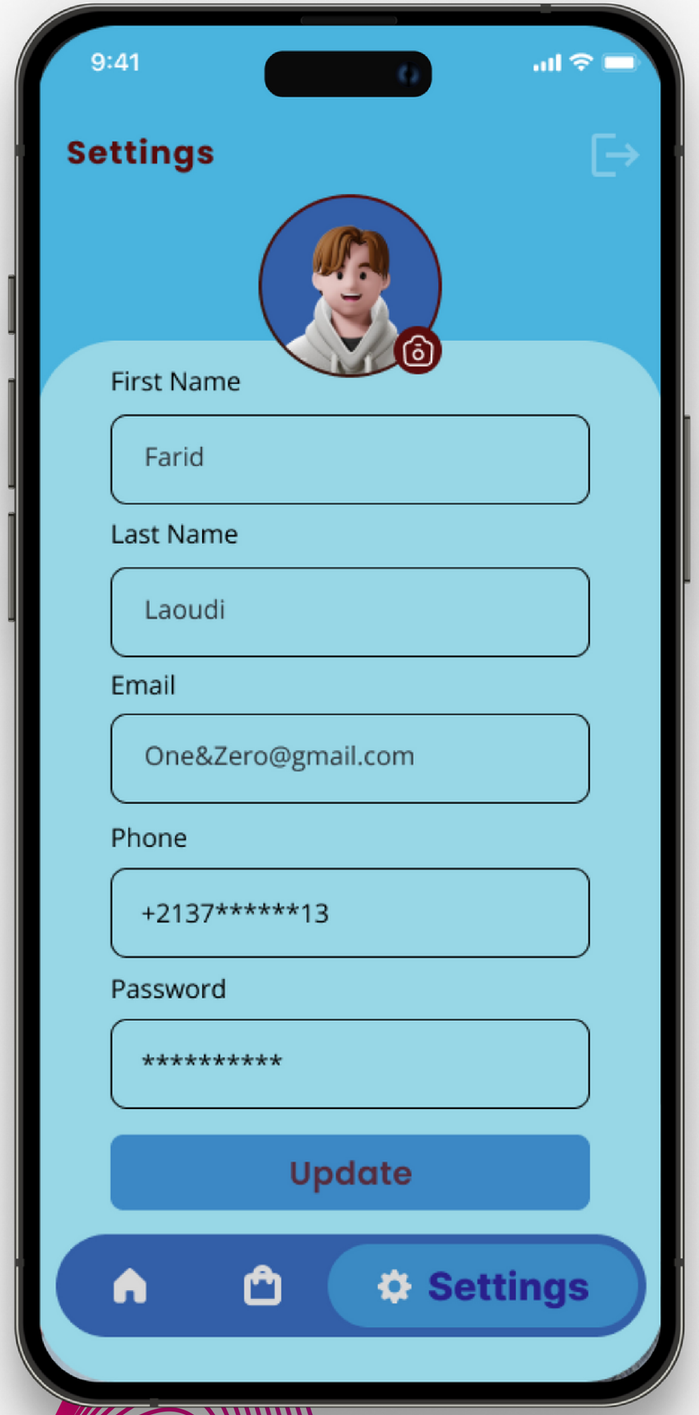

9:41
Settings
First Name
Farid
Last Name
Laoudi
Email
One&Zero@gmail.com
Phone
+2137******13
Password
****
Update
Settings

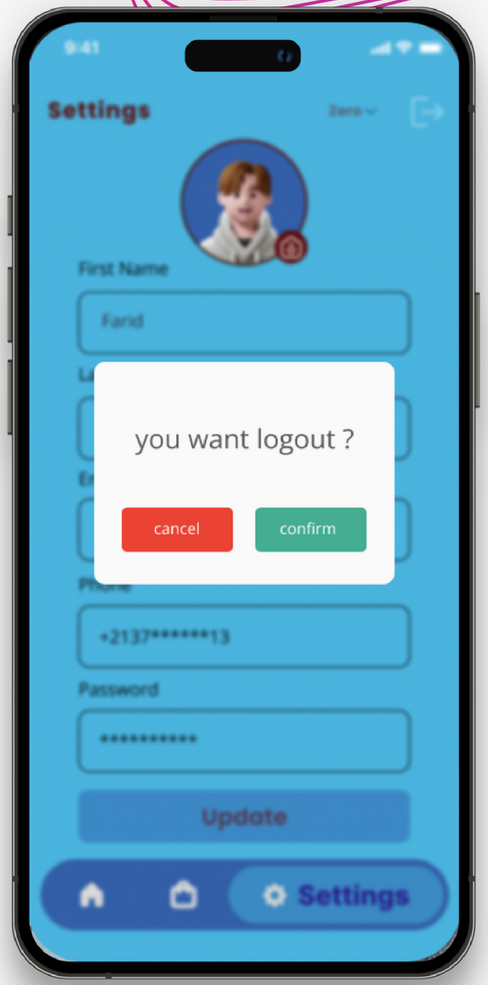

9:41
Home
Zero
Total weight
700 kg
Total profit
60000.00 DA
Most used items
Shoes
Banana
Lamp
Total weight
Total profit
Banana
Boxes : 50
700 kg
60000.00 DA
Lamp
Boxes : 33
100 kg
25000.00 DA
Shoes
Boxes : 20
400 kg
45000.00 DA
Home


9:41
Items
Zero
Search for an item
Item name
Weight
Profit
Banana
60 kg
7000.00 DA
Lamp
60 kg
7000.00 DA
Shoes
60 kg
7000.00 DA
Car
60 kg
7000.00 DA
Clothes
60 kg
7000.00 DA
Lamp
60 kg
7000.00 DA
Select the available weight
Generate
Add item
Items


9:41
Settings
First Name
Farid
Last Name
Laoudi
Email
One&Zero@gmail.com
Phone
+2137******13
Password
****
Update
Settings

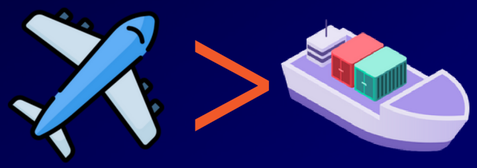

9:41
Home
Zero
Total weight
700 kg
Total profit
60000.00 DA
Most used items
Shoes
Banana
Lamp
Total weight
Total profit
Banana
Boxes : 50
700 kg
60000.00 DA
Lamp
Boxes : 33
100 kg
25000.00 DA
Shoes
Boxes : 20
400 kg
45000.00 DA
Home


9:41
Settings
First Name
Farid
Last Name
Laoudi
Email
One&Zero@gmail.com
Phone
+2137******13
Password
****
Update
Settings


Settings
First Name
Farid
you want logout ?
cancel
confirm
+2137******13
Password
****
Update
Settings

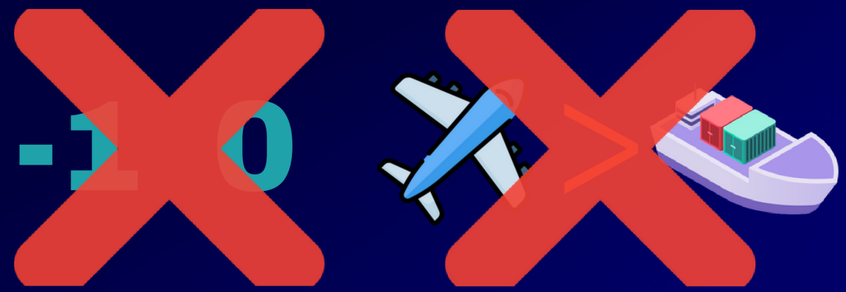
The Algorithme

- preprocessing
- To remove outliers (kg = 0, negative values, NaN values).

-1 0 

The Algorithme

- preprocessing
- To remove outliers (kg = 0, negative values, NaN values).



The Algorithme

- preprocessing
- To remove outliers (kg = 0, negative values, NaN values).
- sort the items by the ratio value/weight
- take the maximum of each time withe highest ratio

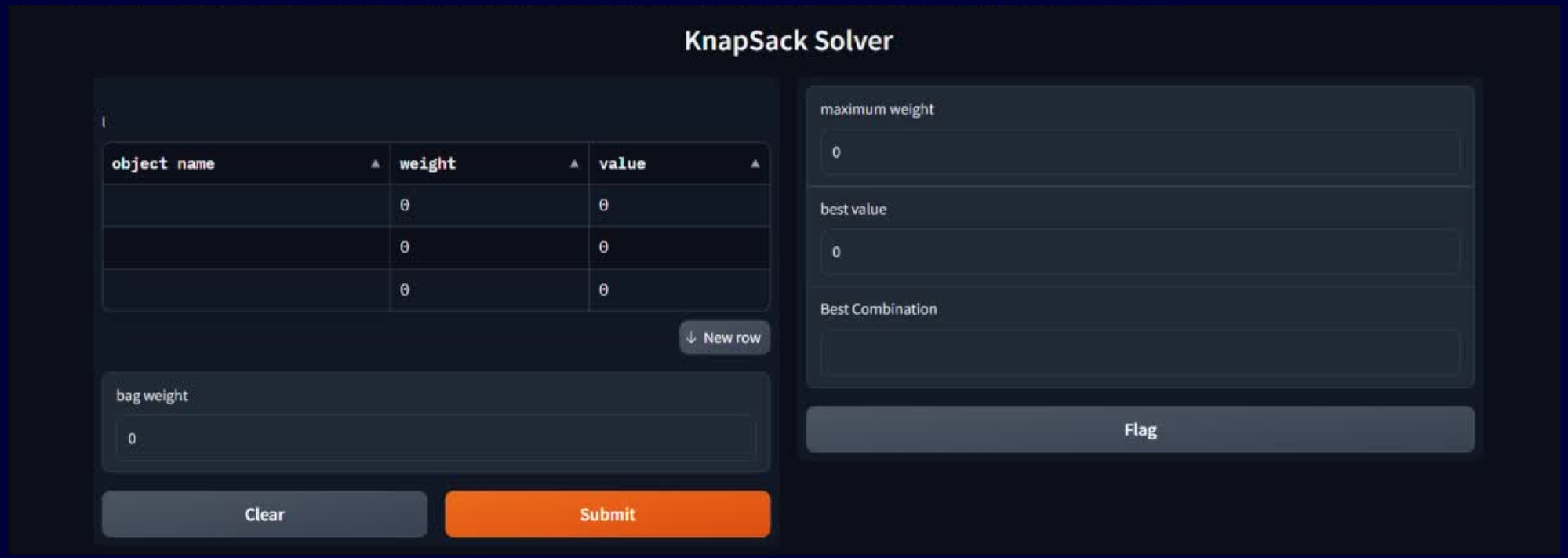
The Algorithme

- the code filters out the items .
- The loop calculates the value/weight .
- The items are sorted by value/weight descending order .
- variables k, h, and o to keep track of the total value, total weight, and the selected items in the knapsack.
- The second loop calculates how many of each item can fit based on the remaining weight limit.
- The function returns the total weight h, total value k, and string o containing the selected items and their quantities.

```
def tri(l,W):
    l.dropna
    l= l[l.weight <=W]
    l= l[l.weight >0]
    l=l[l.value >0]
    for i in range(0,len(l.index)):
        l["value"].loc[l.index[i]]=l["value"].loc[l.index[i]]//l["weight"].loc[l.index[i]]
    l=l.sort_values(by=['value'], ascending=False)
    k=0
    h=0
    o=""
    for i in range(0,len(l.index)):
        k=k+l["weight"].loc[l.index[i]]*(W//l["weight"].loc[l.index[i]])*l["value"].loc[l.index[i]]
        h=h+l["weight"].loc[l.index[i]]*(W//l["weight"].loc[l.index[i]])
        o=o+str(W//l["weight"].loc[l.index[i]])+l["object name"].loc[l.index[i]]+" "
        W=W-l["weight"].loc[l.index[i]]*(W//l["weight"].loc[l.index[i]])
    return h,k,o
```

The Algorithme

Test with Gradio



Thanks !