AoA Stimulation User Guide

Sum of Subsets Problem – Backtracking Approach

Group Members:

- **16010123012** – Aaryan Sharma
- **16010123017** – Aditey Kshirsagar
- **16010123023** – Aditya Baheti

Division: A
Batch: A1

Website: https://sum-of-subsets.vercel.app/

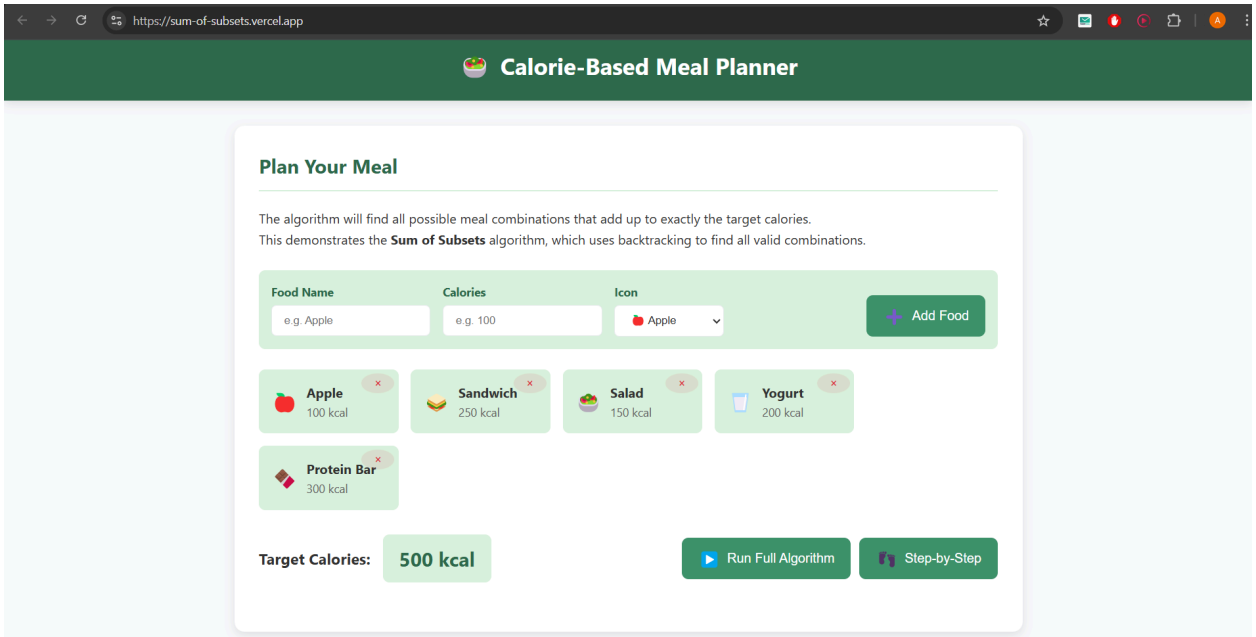
User Guide Documentation

1. SIMULATION OVERVIEW

This interactive visualization simulates the **Sum of Subsets Problem** using the **Backtracking Algorithm**. The objective is to find all subsets of given numbers that sum up to a target value.

Users can:

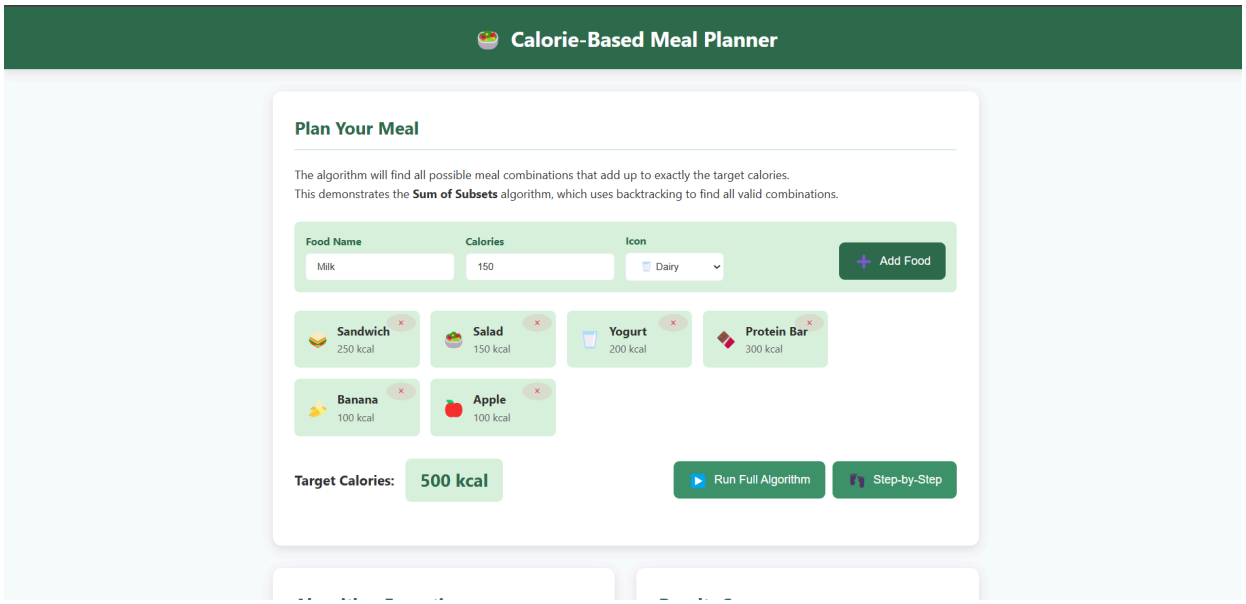
- Define the input set and target sum
- Step through the backtracking process
- Visually follow how subsets are explored
- View feasible solutions in real-time



2. KEY FEATURES

Problem Customization:

- Define your own input set (up to 10 elements)
- Enter a custom target sum
- Clear and reset input for new simulations



Visualization Features:

- Step-by-step subset generation
- Highlighting current subset being evaluated
- Feasibility check animation for each subset
- Clear indicators for valid/invalid combinations

Controls:

- **Start**: Begin the simulation
- **Step**: Move through each recursive call manually
- **Prev/Next**: View the previous or next subset of the particular subset currently being checked.

Algorithm Execution

Execution Log

Exploring index 0, subset: [], sum: 0

Exploring index 1, subset: [Sandwich], sum: 250

Exploring index 2, subset: [Sandwich,Salad], sum: 400

Exceeded target (sum = 600 > 500)

Exceeded target (sum = 700 > 500)

Valid plan found: [Sandwich,Salad,Banana]

Exploring index 3, subset: [Sandwich,Yogurt], sum: 450

Exceeded target (sum = 750 > 500)

Previous | Next Step

Results Summary

24	2	7
Total Recursive Calls	Valid Meal Plans	Pruned Paths

Valid Meal Combinations

1. Sandwich + Salad + Banana
   500 kcal
2. Yogurt + Protein Bar
   500 kcal

Target Calories: 500 kcal

Run Full Algorithm | Step-by-Step

Algorithm Execution

Step-by-Step Execution

Exploring index 0, subset: [], sum: 0

Exploring index 1, subset: [Sandwich], sum: 250

Exploring index 2, subset: [Sandwich,Salad], sum: 400

Previous | Next Step

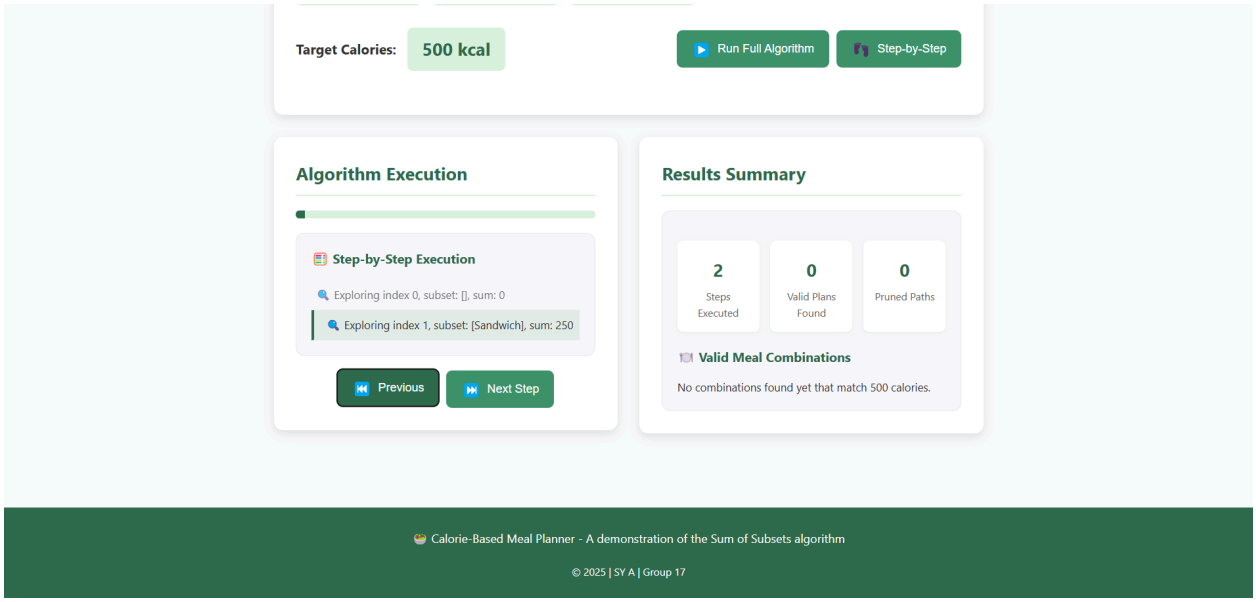
Results Summary

3	0	0
Steps Executed	Valid Plans Found	Pruned Paths

Valid Meal Combinations

No combinations found yet that match 500 calories.

Calorie-Based Meal Planner - A demonstration of the Sum of Subsets algorithm



Solution Display:

- Valid subsets displayed in output section
- Real-time subset value display
- Count of valid solutions found
- Total recursive calls shown

Results Summary

64	8	19
Total Recursive Calls	Valid Meal Plans	Pruned Paths

🍽️ Valid Meal Combinations

1. 🥪 Sandwich + 🥗 Salad + 🍌 Banana
   500 kcal
2. 🥪 Sandwich + 🥗 Salad + 🍎 Apple
   500 kcal
3. 🥪 Sandwich + 🍌 Banana + 🥛 Milk
   500 kcal

Results Summary

3. 🥪 Sandwich + 🍌 Banana + 🥛 Milk
   500 kcal
4. 🥪 Sandwich + 🍎 Apple + 🥛 Milk
   500 kcal
5. 🥗 Salad + 🥛 Yogurt + 🥛 Milk
   500 kcal
6. 🥗 Salad + 🍌 Banana + 🍎 Apple + 🥛 Milk
   500 kcal
7. 🥛 Yogurt + 🍫 Protein Bar
   500 kcal
8. 🍫 Protein Bar + 🍌 Banana + 🍎 Apple
   500 kcal

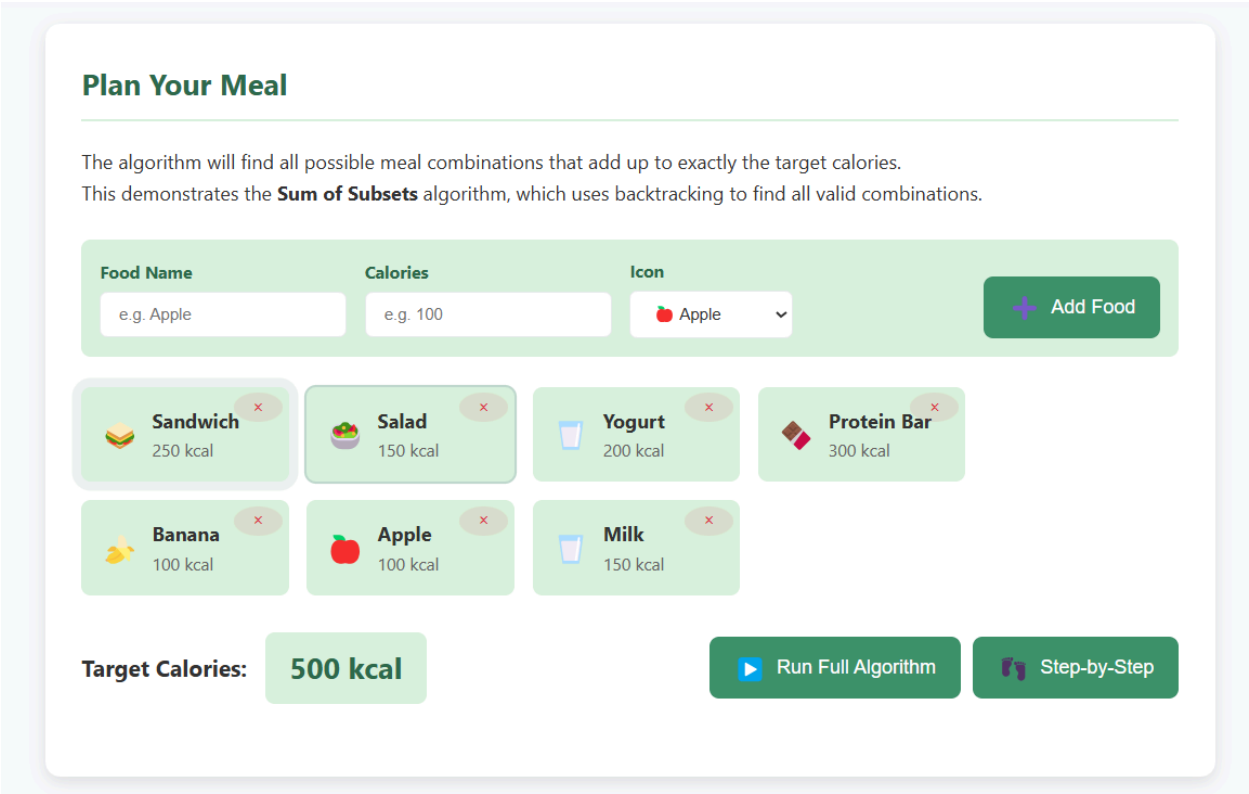
3. HOW IT WORKS

Algorithm Flow:

1. **Initialization**:
    - User enters set and target sum
    - System initializes recursive backtracking function
2. **Backtracking Execution**:
    - Recursively include/exclude each element
    - Keep track of running sum
    - Store subset if it matches the target
3. **Visualization**:
    - Current subset is highlighted
    - Real-time sum calculation
    - Valid subsets shown as output
    - Visual transition for decisions

User Interaction:

- Input set & target via form
- Control buttons for simulation flow
- Output updates as subset checks are completed


Plan Your Meal
The algorithm will find all possible meal combinations that add up to exactly the target calories.
This demonstrates the Sum of Subsets algorithm, which uses backtracking to find all valid combinations.
Food Name
e.g. Apple
Calories
e.g. 100
Icon
Apple
Add Food
Sandwich
250 kcal
Salad
150 kcal
Yogurt
200 kcal
Protein Bar
300 kcal
Banana
100 kcal
Apple
100 kcal
Milk
150 kcal
Target Calories:
500 kcal
Run Full Algorithm
Step-by-Step

4. TECHNICAL ASSUMPTIONS

- Input array size ≤ 10
- Minimum Calories is 100kcal.
- Target sum and elements must be integers
- Only unique subsets shown (no duplicates)
- UI optimized for desktop resolutions
- Chrome/Edge browsers recommended

5. IMPLEMENTATION DETAILS

Technologies Used:

- **HTML/CSS** for structure, style and responsiveness
- **JavaScript** for algorithm logic

Performance Considerations:

- Complexity: $O(2^n)$ subset generation
- Optimized rendering with React state management
- Recursive visual steps avoid blocking UI
- Memory-efficient subset tracking