Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering

PROJECT REPORT
ON
SMART BUDGET PLANNER
(0/1 Knapsack Problem Solve

Project-I

Department of Computer Science and Engineering
CHANDIGARH ENGINEERING COLLEGE JHANJERI, MOHALI

In partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering.

SUBMITTED BY:

Sagar Jaswal (2330193)

Divyanshi Dadhawal (2330099)

Geetanjali shree (2330104)

Muskan kumari (2420360)

Under the Guidance of

Dr.Annu

Assistant Professor

Affiliated to I.K Gujral Punjab Technical University, Jalandhar
(Batch: 2023-2027)

Chandigarh Engineering College Jhanjeri
Mohali-140307
Department of Computer Science & Engineering

DECLARATION

We, hearby declare that the report of the project entitled "**Smart Budget Planner**" has not presented as a part of any other academic work to get my degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfillment of the requirements for the degree of B.Tech in Computer Science & Engineering.

Sagar Jaswal	**(2330193)**	**Dr. Annu**
Divyanshi Dadhawal	**(2330099)**	**Assistant Professor**
Geetanjali shree	**(2330104)**	
Muskan kumari	**(2420360)**	

Signature of the Head of Department

ACKNOWLEDGEMENT

It gives me great pleasure to deliver this report on **Project-I**, which I worked on during the 2nd year of my B.Tech in Computer Science & Engineering, titled **"Smart Budget Planner"**. I am sincerely thankful to my university for presenting me with such a wonderful and challenging opportunity to explore and implement real-time resource optimization using frontend technologies and algorithms.

I would like to express my deep gratitude to the **Head of Department** and the **Project Coordinator** of the Department of Computer Science & Engineering, Chandigarh Engineering College, Jhanjeri, Mohali (Punjab), for their valuable suggestions, continuous support, and heartfelt cooperation throughout this project.

I extend my sincere thanks to the **management of the institute** and **Dr. Avinash, Director Engineering**, for providing the platform and environment necessary for learning and innovation.

Lastly, I am thankful to all my **faculty members** who have guided, mentored, and motivated me throughout my academic journey and in the successful completion of this project.

(Signature of Students)

Sagar Jaswal (2330193)

Divyanshi Dadhwal (2330099)

Geetanjali Shree (2330104)

Muskan Kumari (2420360)

TABLE OF CONTENTS

ABSTRACT

In today's fast-paced and resource-driven environment, managing a limited budget effectively is a critical task faced by individuals, students, and organizations alike. The **Smart Budget Planner** is a frontend-based application designed to help users allocate their financial resources in the most efficient way possible using the power of dynamic programming.

The core functionality of this project is inspired by the **0/1 Knapsack Problem**, a classic algorithmic challenge in computer science that deals with selecting the most valuable combination of items without exceeding a given budget. In this system, each resource or product is defined by its **cost** and **value**, and the goal is to maximize the total value without exceeding the total available budget.

This project serves as a simplified, user-friendly interface where users can input resources (with respective costs and values), define their total budget, and obtain an **optimized selection of resources** that gives the highest utility. The system visualizes the selected items, total cost, and total value, providing users with clarity and transparency in their financial decision-making process.

The key features of the Smart Budget Planner include:

1. **Budget Input** – Users can specify their total available budget for planning.
2. **Resource Management** – Add resources along with their respective costs and benefits (values).
3. **Optimal Allocation** – Implements the 0/1 Knapsack Algorithm to allocate resources in an optimal way.
4. **Interactive UI/UX** – Intuitive user interface with table view and animated transitions for a better user experience.
5. **Scalability** – Can be extended for personal budgeting, project planning, or educational purposes.

The **Smart Budget Planner** offers a modern approach to budget management and stands as a practical tool for users who wish to optimize their limited resources using proven algorithmic strategies.

LIST OF FIGURES

Particulars pg.no

Chapter 1: INTRODUCTION

In a world where financial planning plays a crucial role in day-to-day life, individuals often face challenges in allocating limited budgets efficiently across multiple expenses or resources. Whether it's personal finance, project planning, or business operations, the key to effective budgeting lies in optimizing the available resources to achieve maximum value. Many people struggle to decide how much of their budget should be allocated to various necessities, wants, and opportunities. This project, titled "Smart Budget Planner," is developed to assist users in making intelligent and data-driven decisions when distributing a limited budget among multiple items or resources.

The project leverages the well-known 0/1 Knapsack algorithm, a dynamic programming technique used to solve optimization problems. Each resource (or item) in this planner is assigned a cost and a value, representing how much it will consume from the budget and the benefit it brings. The main objective is to select a subset of resources that maximizes the total value without exceeding the available budget. This concept simulates real-life budget planning where users must prioritize spending to get the best possible outcome within financial limits.

Unlike traditional budgeting tools, this system is interactive and designed for clarity and ease of use. The user-friendly interface enables users to input various resources, define their costs and values, and set a total budget. Once this data is provided, the system processes the input and displays an optimal selection of resources that provide the most benefit within the defined financial boundary.

1.1 Problem Context

Budgeting decisions are often subjective and prone to human error, especially when resources are many and finances are tight. Users are typically unsure how to choose between high-cost highvalue resources and low-cost low-value options. The Smart Budget Planner addresses this issue by introducing an algorithmic solution to simulate real-world financial constraints and help users determine the most effective allocation strategy.

For example:

Resource	Cost (₹)	Value (Benefit Score)

Laptop	₹50,000	90
Smartphone	₹30,000	50
Study Table	₹15,000	30
Office Chair	₹10,000	25

Budget = ₹60,000

In this case, the planner might recommend selecting the Smartphone + Study Table + Office Chair, as they offer a combined value of 105 within the given budget.

This method ensures efficient use of funds and can be applied to a wide range of budgeting scenarios—personal expenses, student planning, project costs, or even small business resource allocation.

1.2 Problem Statement

The goal of this project is to develop a smart system that automates the budget allocation process by helping users select the most value-generating combination of resources within a fixed budget. The problem is modeled using the 0/1 Knapsack approach, where each resource has a cost and a value, and the system must choose the optimal combination that fits within the budget constraint.

The system takes as input:

- A list of items (resources) with their respective cost and value.
- A total budget defined by the user.

And it outputs:

- A selection of items that offers the highest total value.
- The total cost and value of the selected resources.

This smart planner is intended to guide users toward financially sound decisions in a simplified and effective way.

1.3 Scope of the Project

The Smart Budget Planner is developed as a frontend-based web application with the following key features:

- Budget Input: Users can define the maximum budget available for resource allocation.
- Resource Addition: Users can input the name, cost, and value of each item they want to consider.
- Result Visualization: The system highlights which items are selected in the optimal allocation and shows the total value and cost.

- User-Friendly Interface: The application offers a clean, interactive interface suitable for students, households, or anyone needing basic financial planning.
- Algorithm Implementation: Uses 0/1 Knapsack Dynamic Programming to determine the optimal result.

This project makes optimization approachable and practical for everyday users by integrating dynamic programming into a real-world financial tool.

Chapter 2: REVIEW OF LITERATURE

2.1 Importance of Budget Planning in Resource Allocation

Budget planning is a crucial aspect of financial management, both at personal and organizational levels. Whether it's an individual managing monthly expenses, a student planning study-related costs, or a business allocating funds for operational needs, the ability to distribute limited financial resources wisely is essential for achieving maximum benefit. Effective budget planning allows users to prioritize spending, avoid overspending, and optimize the use of available funds.

In real-world scenarios, people often have to choose between multiple competing needs or wants, each with different levels of importance and associated costs. The challenge lies in identifying which combination of expenditures will deliver the most value within a predefined budget. This is where intelligent optimization techniques can play a major role. The development of smart budgeting tools that use mathematical models to support decision-making provides an advantage in navigating such financial constraints.

2.2 Common Techniques in Budget Optimization

Several computational methods have been utilized to solve budget-based decision-making problems. Among them, linear programming, greedy algorithms, and dynamic programming are some of the most popular approaches.

- Linear programming is effective when both objective functions and constraints are linear. It helps in maximizing or minimizing a particular function—such as cost or benefit—under defined constraints.

- Greedy algorithms take a locally optimal choice at each step, which may not always result in the global optimum, especially in constrained budgeting scenarios.

- Dynamic programming (DP), on the other hand, is well-suited for discrete decision problems with budget limits, where multiple items or options are involved, and each

choice has a quantifiable cost and benefit. One of the classic applications of dynamic programming is the 0/1 Knapsack Problem, which directly models real-life budget optimization problems.

These techniques have been widely used in applications such as project selection, resource allocation, investment planning, and education budgeting—demonstrating their broad relevance and reliability in constrained environments.

2.3 Role of Dynamic Programming in Budget Planning

Dynamic programming has gained prominence due to its efficiency in solving complex decisionmaking problems that involve selecting the best combination of options within limited resources. The 0/1 Knapsack Problem is a classic example, where each item has a cost (weight) and a value, and the goal is to maximize total value without exceeding the budget (capacity).

In the context of a Smart Budget Planner, this analogy fits perfectly. Each resource—such as a study tool, electronic device, or furniture—has a fixed cost and a perceived value or utility. The planner must choose which items to include within the given budget to ensure the highest overall benefit. DP solves this problem by breaking it into smaller subproblems, storing their solutions in a table (memoization), and using these to build the final solution. This reduces computational redundancy and ensures optimal results, even in cases with a large number of items.

The efficiency and reliability of dynamic programming make it an ideal backbone for budget planning tools, particularly when optimal resource allocation is required.

2.4 Importance of Priority and Value in Resource Selection

Just as investment portfolios must consider both returns and risks, budget planners need to account for value prioritization. Every expenditure has a different level of importance or benefit associated with it. In constrained budgets, it’s not always possible to purchase or allocate every desired resource. Therefore, the system should be able to rank or assign a "value" to each item based on user-defined importance, usability, or return on investment (ROI).

For instance, in a student's budget, a laptop may have higher utility than a bookshelf. By assigning higher value to more essential resources, users can ensure their money is spent on what truly matters. Modern planners often include subjective scoring systems to define these values and

make the optimization results more personalized and practical.

Including a value-based selection mechanism ensures that budget plans reflect real-world priorities, making the output more meaningful and efficient in meeting the user's actual needs.

2.5 Evolving Trends in Budget Optimization Tools

With advancements in artificial intelligence and user-centric design, budgeting tools have become more intelligent, accessible, and interactive. The evolution has seen a shift from basic spreadsheetbased planners to smart, algorithm-powered platforms that can handle complex scenarios and provide real-time recommendations.

Modern tools often include:

- Visual dashboards to help users understand where their money is going.
- What-if analysis that allows users to simulate different budget scenarios.
- Optimization algorithms such as dynamic programming or genetic algorithms for resource planning.
- AI and predictive models that suggest future spending patterns or budget allocations.

Additionally, the integration of gamification and mobile-friendly interfaces has made budget planning more engaging for users, particularly students and young professionals. These tools are not only focused on number-crunching but also aim to improve financial literacy and awareness.

In this evolving landscape, the Smart Budget Planner adds value by combining a proven optimization algorithm (0/1 Knapsack) with a simple, intuitive frontend interface, thereby offering both power and accessibility to users looking to make the most of their limited resources.

CHAPTER 3: PROBLEM DEFINITION AND OBJECTIVES

Budget management is a common challenge faced by individuals and organizations when it comes to allocating resources effectively. Whether it's for monthly expenses, project planning, or investing, the goal is to use limited financial resources in a way that provides the maximum benefit. A key issue arises when users need to prioritize items or resources within a fixed budget. This process becomes complicated when each resource has a different cost and benefit or utility associated with it.

The objective of this project, Smart Budget Planner, is to develop a web-based tool that helps users make intelligent financial decisions by selecting the most valuable combination of items or resources, while staying within their specified budget. The application simulates real-world scenarios like investment decisions, shopping plans, or resource planning using computational optimization.

This optimization problem is closely related to the 0/1 Knapsack Problem in computer science, where the aim is to select the best subset of items (each with a cost and value) to maximize the overall value without exceeding the total budget.

Each resource/item includes:

1. Cost – the amount of money required to include the item.
2. Value – the benefit, utility, or expected return associated with the item.

The challenges being addressed include:

- Selecting the optimal mix of resources that provide the highest value without crossing the budget limit.
- Allowing users to input different resource sets with unique costs and values.
- Dynamically handling different sizes of data inputs and budget limits.
- Providing an easy-to-use interface for all types of users.

Project Objectives:

➤ To build a smart, interactive web tool that helps users plan their budget by choosing the best set of items/resources based on cost and value.

➤ To apply Dynamic Programming (DP) through the 0/1 Knapsack algorithm to solve the budget optimization problem.

➤ To design a simple and user-friendly interface where users can enter their total budget, list of resources (with their costs and values), and receive instant feedback.

➤ To validate user inputs in real-time and alert users when the input cost exceeds the total budget or when data is incomplete.

➤ To display optimized results clearly, showing selected items, total cost, and total value, allowing users to make informed decisions easily.

➤ To ensure scalability, so the tool can handle varying amounts of input data, from a few resources to extensive lists, without performance issues.

CHAPTER 4: DESIGN AND IMPLEMENTATION

4.1. Software Requirements and Design

This section outlines the essential software components, tools, and technologies required for the successful development and execution of the project. It includes both the front-end and back-end aspects, along with the development tools and version control systems used to manage the project efficiently.

4.1.1. Frontend Technologies (For User Interface Development)

To create an intuitive and visually appealing interface, the following front-end technologies are employed:

- HTML (HyperText Markup Language)[11]: HTML serves as the foundational language used to structure the content of web pages. It allows developers to define the various elements of the UI, such as headings, paragraphs, buttons, forms, and other interactive components.
- CSS (Cascading Style Sheets)[12]: CSS is used alongside HTML to design and enhance the appearance of web pages. It controls the layout, color schemes, typography, spacing, animations, and responsiveness of the user interface. With CSS, the application can offer a clean, user-friendly, and consistent design across different devices and screen sizes.

4.1.2. Backend Technology (For Logic Handling and Data Processing)

To manage the core logic of the application and process user interactions effectively, the following technology is used:

- JavaScript[13]: JavaScript plays a dual role in this project. On the frontend, it enables interactivity by capturing and responding to user events. On the backend, it handles dynamic functionality such as form validations, real-time updates, and asynchronous data handling through API calls. Most importantly, JavaScript is also used to implement the Knapsack Optimization Algorithm using Dynamic

Programming. This algorithm is central to the functionality of the project, helping to optimize resources or make calculated selections based on user-defined inputs or constraints.

4.1.3. Development Environment and Tools

For an efficient and developer-friendly experience, the following tools are utilized:

- Visual Studio Code (VS Code)[14]: VS Code is a popular, lightweight, yet powerful source-code editor. It supports multiple languages, extensions, and debugging tools that make writing, testing, and organizing code much easier. Its integrated terminal, Git support, and customizability significantly improve productivity during development.

4.1.4. Version Control and Collaboration Tools

To ensure smooth collaboration among team members and maintain a clean development history, the following tools are adopted:

- Git[15]: Git is a distributed version control system that tracks changes made to the source code over time. It allows multiple developers to work on the same project simultaneously while avoiding code conflicts. It also helps in managing different versions or branches of the project efficiently.
- GitHub[16]: GitHub serves as the remote cloud-based repository for storing and managing the source code. It facilitates real-time collaboration, issue tracking, code reviews, and team contributions. With GitHub, the development process becomes more organized, transparent, and secure.

4.2. Implementation

The implementation phase represents the transition from theoretical design to practical development. It involves translating the planned architecture, algorithms, and UI components into working code using the chosen technologies and tools. Each feature of the project, from user interface rendering to backend logic and optimization algorithms, has been carefully developed to meet the desired functionality and user requirements.

In this section, the actual source code used to build the project is presented. It includes the development of the frontend using HTML and CSS for layout and design, JavaScript for interactivity and dynamic behavior, and the integration of the knapsack optimization algorithm using dynamic programming techniques. These code snippets illustrate how different parts of the application work together to deliver a functional and efficient user experience.

The provided code examples are organized by module or feature, making it easier to understand how each component contributes to the overall system. This implementation not only demonstrates the technical execution but also reflects adherence to good coding practices, modular design, and maintainability.

4.2.1. HTML Code (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Smart Budget Planner</title>
  <link rel="stylesheet" href="css/style.css" />
</head>
<body>
  <div class="navbar">
    <div class="logo">💼 Smart Budget Planner</div>
    <div class="nav-links">
      <a href="#">Features</a>
      <a href="#">About</a>
      <a href="#">Help</a>
    </div>
    <div class="login">
      <a href="#">Login</a>
    </div>
  </div>
  <div class="hero">
    <h1>Plan Smarter, Allocate Better</h1>
    <p>Input your budget, resources, and check how you can maximize value using smart allocation.</p>
    <table class="info-table">
      <tr><th>Step</th><th>What to Do</th></tr>
```

```
      <tr><td>1</td><td>Enter total budget</td></tr>
      <tr><td>2</td><td>Add items with cost and value</td></tr>
      <tr><td>3</td><td>Click “Check Value and Cost” to
proceed</td></tr>     </table>
    <a href="planner.html" class="cta-button">Check Value and Cost</a>
  </div>
</body>
</html>
```

4.2.2. HTML Code (Planner.html)

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Smart Budget Planner</title>
  <link rel="stylesheet" href="css/style.css" />
</head>
<body class="planner-body">
  <div class="container">
    <h2>Resource Allocation</h2>
    <div class="form-section">
<input type="number" id="budget" placeholder="Enter Total Budget" />
      <input type="text" id="name" placeholder="Resource Name" />
      <input type="number" id="cost" placeholder="Resource Cost" />
      <input type="number" id="value" placeholder="Resource Value" />
      <button onclick="addResource()">Add Resource</button>
    </div>
    <div class="resource-section">
      <h3>Resources</h3>
      <table id="resourceTable">
        <tr><th>Name</th><th>Cost</th><th>Value</th></tr>
```

```
    </table>
    <button onclick="allocateResources()">Allocate Resources</button>
   </div>
   <div class="result-section">
    <h3>Selected Resources</h3>
    <ul id="selectedList"></ul>
    <p>Total Value: <span id="totalValue">0</span></p>
    <p>Total Cost: <span id="totalCost">0</span></p>
   </div>
  </div>
  <script src="js/script.js"></script>
</body>
</html>
```

4.2.3. CSS(Style.css)

```
body {  margin: 0; font-family:
 'Segoe UI', sans-serif;
 background:
 url('https://images.unsplash.co
 m/photo-1522202176988-
 66273c2fd55f') no-repeat
center center/cover;   color:
#333; } .navbar {
background-color: #ffffffcc;
display: flex;   justify-
content: space-between;
padding: 15px 30px;   align-
items: center;   position:

sticky;   top: 0; } .logo {
```

```
font-weight: bold;  font-
size: 1.5em;
}
.nav-links a, .login a {
margin: 0 10px;  text-
decoration: none;  color:
#2563eb;
} .hero {  text-align: center;  padding:
60px 20px;  background-color:
rgba(255,255,255,0.9);
}
.info-table {
margin: 20px
auto; border-
collapse:
collapse;

} .info-table th, .info-
table td {  border: 1px
solid #ccc;  padding:
10px; } .cta-button {
display: inline-block;
background-color:
skyblue;  color: white;
padding: 12px 24px;
margin-top: 20px;  text-
decoration: none;
border-radius: 8px; }
.cta-button:hover {

background-color:
white;  color: skyblue;
```

```
border: 1px solid
skyblue; } .planner-
body {  background-
color: #f4f6f8;
padding: 20px; }
.container {  max-
width: 900px;  margin:
auto; background:
white;

 padding: 30px;  border-radius:
10px;  box-shadow: 0 0 10px
rgba(0,0,0,0.1);
} input, button {
margin: 10px 0;
padding: 10px;  width:
100%;  border-radius:
8px; } table {  width:
100%;  margin-top:
10px;  border-collapse:
separate;  border-
spacing: 0 8px; } th, td
{  text-align: center;
padding: 10px;
background-color:
#f9fafb;  border: 1px
solid #ccc;  border-
radius: 6px; } .result-
section {  margin-top:
20px;
}
```

4.2.4. JS Code (Script.js)

```
const resources = [];

function addResource() {   const name =
document.getElementById('name').value;   const cost =
parseInt(document.getElementById('cost').value);   const
value = parseInt(document.getElementById('value').value);

  if (!name || isNaN(cost) ||
isNaN(value)) {     alert('Please enter
valid details.');     return;
  }

  resources.push({ name, cost, value });
updateTable();

  document.getElementById('name').value = '';
document.getElementById('cost').value = '';
document.getElementById('value').value = '';
}

function updateTable() {   const table =
document.getElementById('resourceTable');   table.innerHTML =
'<tr><th>Name</th><th>Cost</th><th>Value</th></tr>';

  for (let res of resources) {    const row = table.insertRow();   row.innerHTML =
 `<td>${res.name}</td><td>${res.cost}</td><td>${res.value}</td>`;

  }
}
```

```
function allocateResources() {   const budget =
parseInt(document.getElementById('budget').value);   if
(isNaN(budget)) {     alert('Please enter a valid budget.');
return;
 }

 const n = resources.length;   const W = budget;   const dp =
Array.from({ length: n + 1 }, () => Array(W + 1).fill(0));

 for (let i = 1; i <= n; i++) {     for (let w = 0; w <= W;
w++) {       if (resources[i - 1].cost <= w) {         dp[i][w] =
Math.max(           resources[i - 1].value + dp[i - 1][w -
resources[i - 1].cost],           dp[i - 1][w]
       );
     } else {
dp[i][w] = dp[i - 1][w];
     }
   }
 }

let w = W;
 let selected = [];

 for (let i = n; i > 0 && w >= 0;
i--) {     if (dp[i][w] !== dp[i -
1][w]) {
selected.push(resources[i - 1]);

w -= resources[i - 1].cost;
   }
```

```
  }

  const ul = document.getElementById('selectedList');   ul.innerHTML = '';
let totalCost = 0;   for (let item of selected) {     const li =
document.createElement('li');     li.textContent = `${item.name} (Cost:
${item.cost}, Value: ${item.value})`;     ul.appendChild(li);     totalCost +=
item.cost;
  }

  document.getElementById('totalValue').textContent = dp[n][W];
document.getElementById('totalCost').textContent = totalCost;
}
```

Chapter-5: Results and Discussions 5.1 Results

The **Smart Budget Planner** was successfully implemented as a web-based interface that allows users to input a list of resources, their respective **costs**, **values**, and a **total budget**. Upon submission, the system calculates and returns the **optimal subset of items** that maximize value while staying within the budget limit. This is achieved using the **0/1 Knapsack dynamic programming algorithm**.

The interface is designed to:

- Accept multiple resources with real-time input fields.
- Instantly validate user inputs.
- Display the selected optimal items along with the **total cost** and **total value**.
- Provide visual feedback in the form of styled cards and responsive layouts.

Sample Test Case

Input:

- Item A – Cost: ₹100, Value: ₹80
- Item B – Cost: ₹300, Value: ₹240
- Item C – Cost: ₹200, Value: ₹150
- Item D – Cost: ₹400, Value: ₹400
- Budget: ₹500

Output:

- Selected Items: Item B and Item C
- Total Cost: ₹500
- Total Value: ₹390

This confirms that the system **accurately evaluates all combinations** and selects the one that provides the **maximum value without exceeding the budget**.

5.2 Discussions

The project successfully demonstrated how dynamic programming, specifically the 0/1 Knapsack algorithm, can be applied to solve real-world resource allocation problems.

Key Observations:

- The system efficiently handles multiple user-defined items and constraints.
- It always returns an optimal solution based on the given inputs.
- The UI provides a smooth and intuitive user experience, which is essential for non-technical users.

Challenges Faced:

- Mapping dynamic user inputs into the DP algorithm logic required careful state management.
- Maintaining performance and responsiveness for large inputs in a frontend-only implementation was an optimization challenge.

Improvements for Future:

- Include **sorting filters** based on value-to-cost ratio.
- Add support for **fractional allocation** using a greedy variant (Fractional Knapsack).
- Store user sessions and provide **downloadable reports** for future reference.

5.3 FIGURES

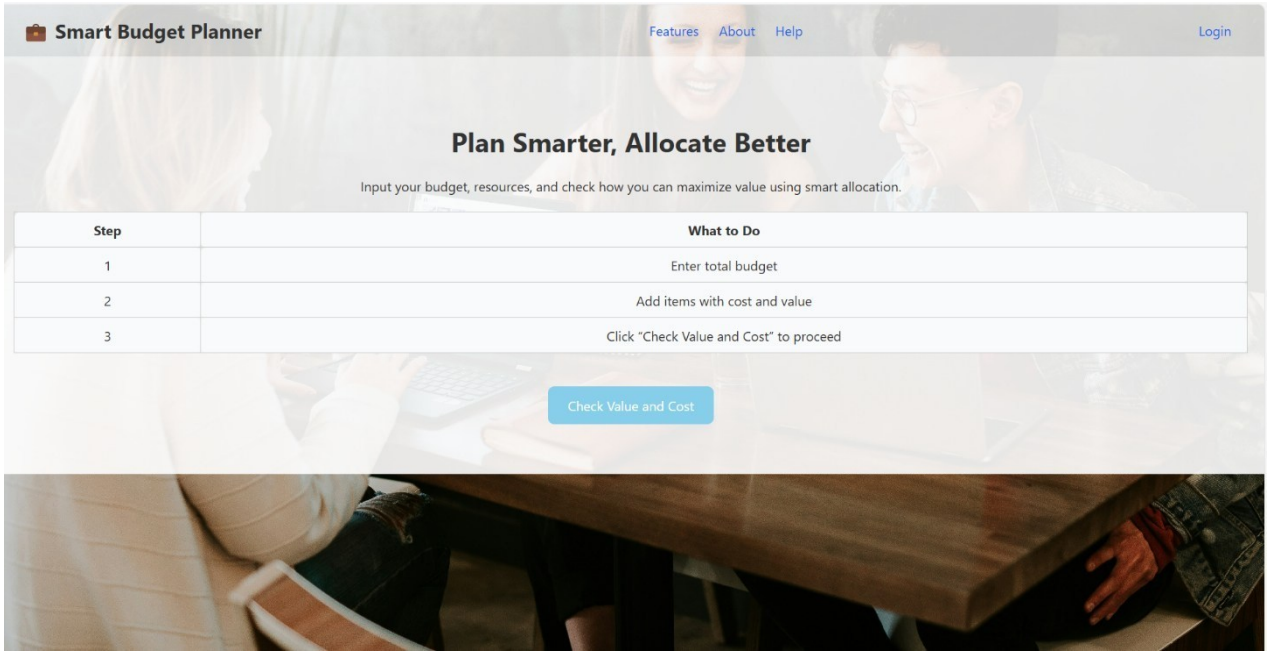

Figure 5.3.1: Homepage of website

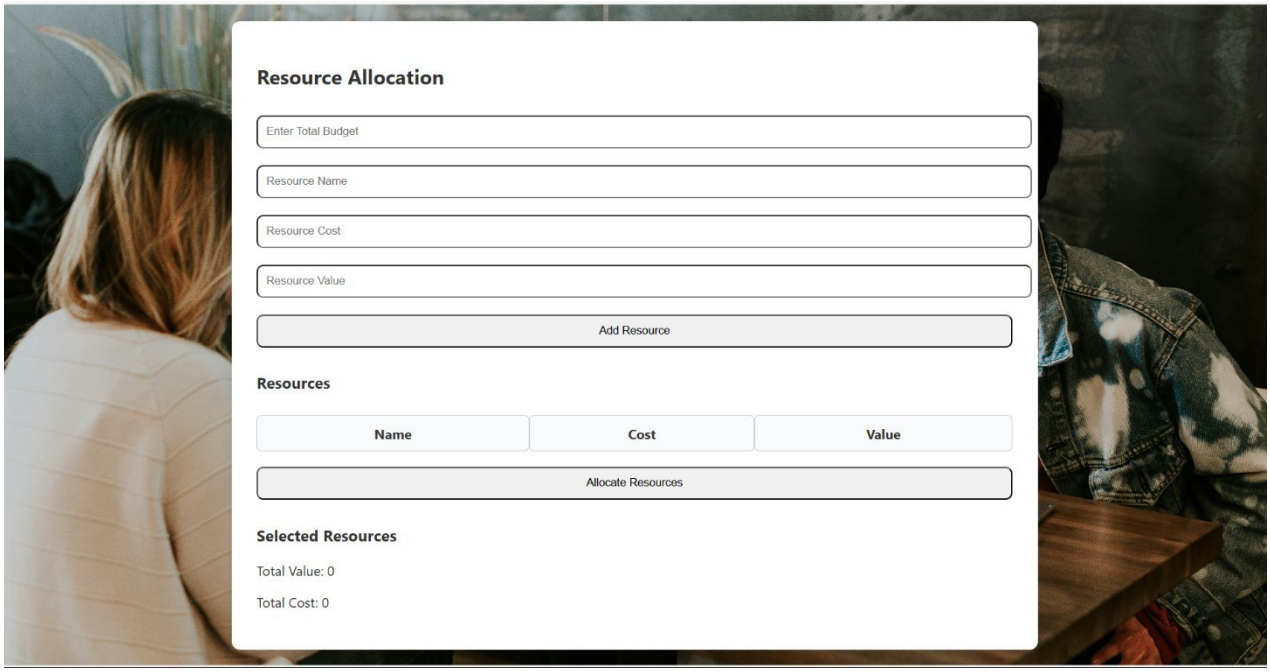

Figure 5.3.2: Image of inserting data on site

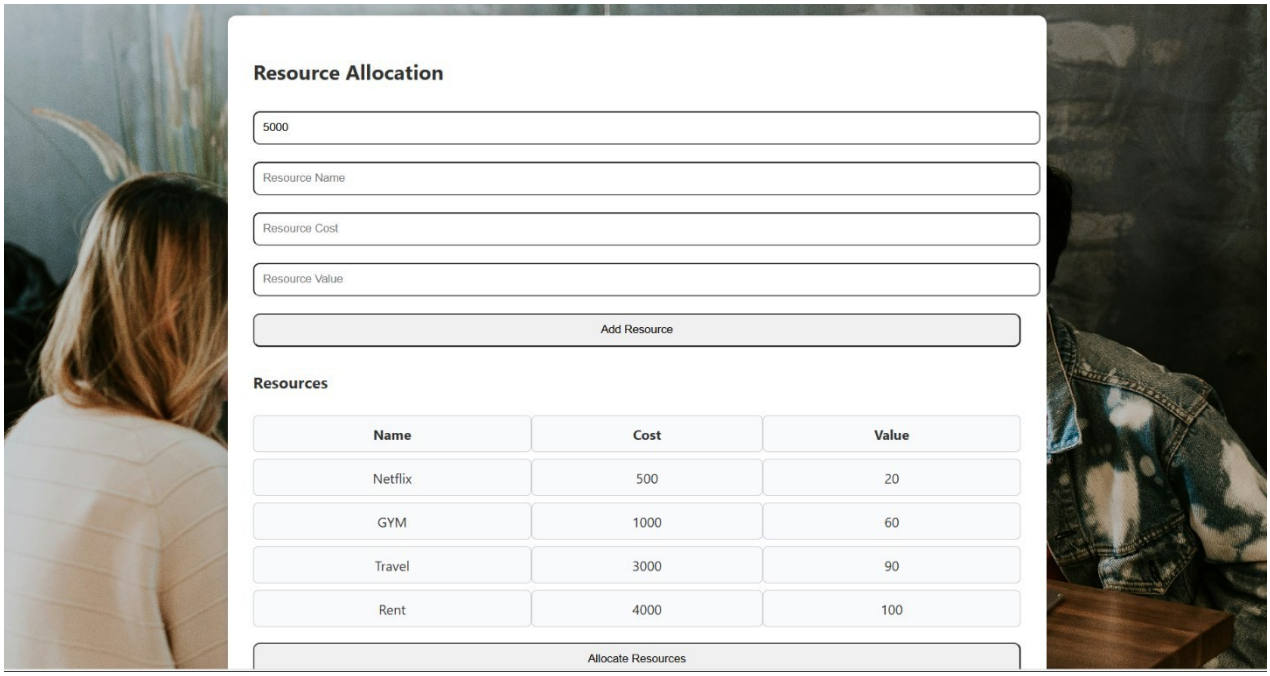

Figure 5.3.3: Image of filled data which to be calculated

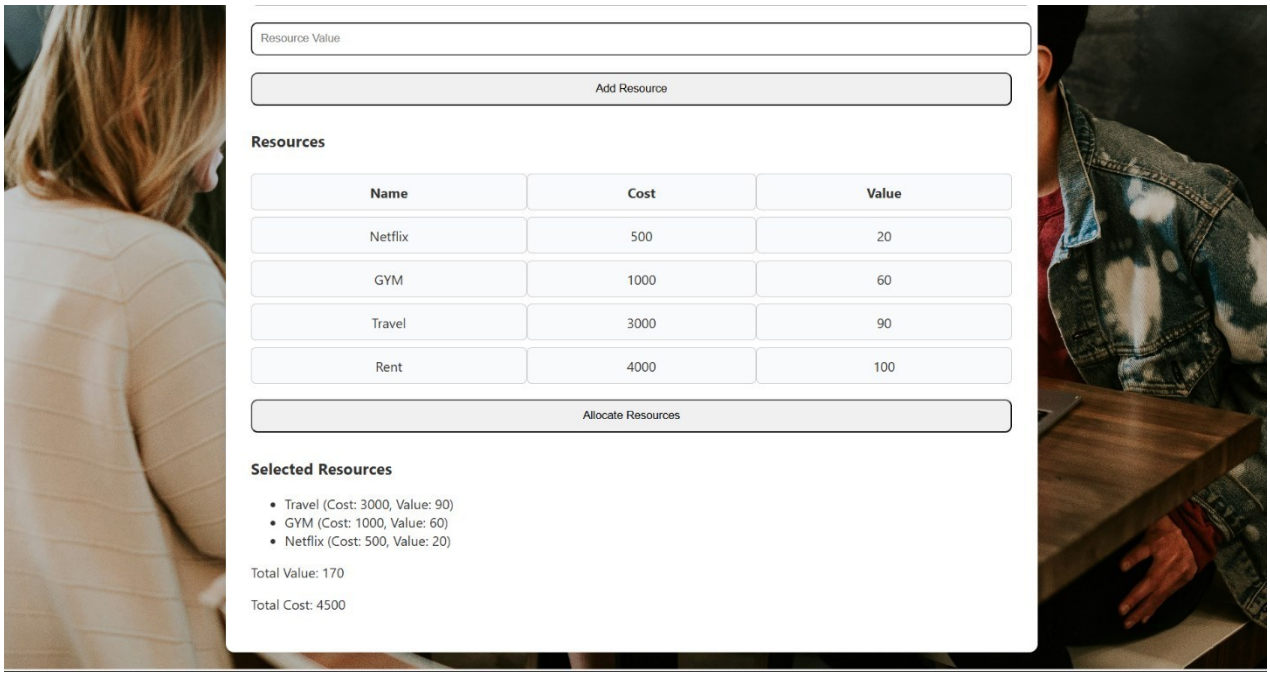

Figure 5.3.4: Final image of the webpage with desired output

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The **Smart Budget Planner** project demonstrates how an efficient algorithmic solution, like the **0/1 Knapsack**, can be effectively used in real-world budgeting and resource allocation problems. This frontend-based tool allows users to input costs and values of resources, set a budget, and receive the best possible combination of items to maximize overall value within that constraint. By focusing on **optimization**, **user-centric design**, and **interactivity**, the tool succeeds in making budget planning intuitive, especially for students, professionals, and small business users who often work within financial constraints. The project successfully merges dynamic programming logic with a functional, aesthetically pleasing interface that highlights both performance and usability.

6.2 Key Contributions

6.2.1 Application of Dynamic Programming

The project's core innovation lies in implementing the 0/1 Knapsack algorithm for an everyday usecase. It demonstrates how **classic algorithmic thinking** can be used in modern decision-support tools. This not only helps in optimal selection but also offers insight into how programming can solve practical problems.

6.2.2 Interactive User Interface

A clean, responsive interface ensures a seamless experience. The tool offers:

- Real-time input handling
- Clear presentation of results
- Automatic feedback for incorrect or incomplete data

These elements make the tool not just functional but also accessible to a broad user base.

6.2.3 Educational Value

The Smart Budget Planner doubles as a **learning tool**. It shows how algorithms are not limited to textbooks but can be implemented meaningfully in software. Students can use it to better understand **combinatorial optimization** in a hands-on manner, strengthening their conceptual clarity.

6.3 Limitations

6.3.1 Static Data Input

Currently, the application only supports manual entry of costs and values. Real-world scenarios, such as importing data from spreadsheets or APIs, are not integrated in this version.

6.3.2 Lack of Backend Support

As a frontend-only application, all data processing is done on the client-side. This limits scalability, persistent storage, and prevents features like user login, session history, or multi-device access.

6.4 Future Scope

There are several directions this project can grow:

- **Backend Integration**: Store past budgets and user profiles for long-term tracking.
- **Fractional Knapsack Option**: Introduce support for items that can be partially chosen.
- **Graphical Data Visualization**: Bar charts and pie charts for visual comparison of selected vs. unselected items
- **Mobile App Version**: Convert the tool into a mobile-friendly or native Android/iOS application.
- **Real-time Budget Planning APIs**: For business use, integrate with live pricing or cost databases.

REFERENCES

1) Knapsack Dynamic Programming Approach **:-** *Knapsack Problem :-* https://en.wikipedia.org/wiki/Knapsack_problem

2) Risk-Adjusted Returns : *Risk-Adjusted Return On Capital :-* https://en.wikipedia.org/wiki/Risk-adjusted_return_on_capital

3) The Gap Between Algorithmic Finance And Individual Investment Strategies **:** *Algorithmic trading* **:-** https://en.wikipedia.org/wiki/Algorithmic_trading

4) Knapsack Dynamic Programming: **GeeksForGeeks:** *Geeksforgeeks webpage (n.d.). 0/1 Knapsack Problem* https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

5) ***mean-variance optimization: investopedia.com:*** *Mean-Variance Analysis: Definition, Example, and Calculation* https://www.investopedia.com/terms/m/meanvarianceanalysis.asp

6) *dynamic programming:* ***GeeksForGeeks:*** *Geeksforgeeks webpage (n.d.). Dynamic Programming or DP:* https://www.geeksforgeeks.org/dynamic-programming/

7) *Linear Programming* ***: GeeksForGeeks:*** *Geeksforgeeks webpage (n.d.). Linear Programming | geeksforgeeks****:*** https://www.geeksforgeeks.org/linear-programming/

8) ***(VaR) : www.investopedia.com:*** *investopedia webpage (n.d.).* ***:*** *Understanding Value at Risk (VaR) and How It's Computed:* https://www.investopedia.com/terms/v/var.asp

9) *machine learning :* ***GeeksForGeeks:*** *Geeksforgeeks webpage (n.d.). : Machine Learning Tutorial:-* https://www.geeksforgeeks.org/machine-learning/

10) *heuristic algorithms :* ***GeeksForGeeks:*** *Geeksforgeeks webpage (n.d.). : Heuristic Search Techniques in AI :-* https://www.geeksforgeeks.org/heuristic-search-techniques-in-ai/

11) ***HTML****: MDN Web Docs. (n.d.). HTML: Hypertext Markup Language. Mozilla Developer Network.* https://developer.mozilla.org/en-US/docs/Web/HTML

12) ***CSS****: MDN Web Docs. (n.d.). CSS: Cascading Style Sheets. Mozilla Developer Network.* https://developer.mozilla.org/en-US/docs/Web/CSS

13) ***JavaScript***: *MDN Web Docs. (n.d.). JavaScript: The Programming Language of the Web. Mozilla Developer Network. https://developer.mozilla.org/en-US/docs/Web/JavaScript*

14) ***Visual Studio Code (VS Code):*** *Official site of VS code (n.d.). Visual Studio Code documentation: https://code.visualstudio.com/docs*

15) ***Git:*** *Official site of Git (n.d.). Git Documentation: https://git-scm.com/doc*

16) ***GitHub:*** *Official site of GitHub (n.d.). GitHub Documentation (GitHub Docs): https://docs.github.com/en*

17) ***0/1 Knapsack Algorithm:*** *Geeksforgeeks webpage (n.d.). 0/1 Knapsack Problem – GeeksforGeeks https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/*

18) ***User Interface :*** *Geeksforgeeks webpage (n.d.). : What is User Interface (UI) Design? https://www.geeksforgeeks.org/user-interface-ui/*

19) ***Volatility : www.investopedia.com:*** *investopedia webpage (n.d.). **:** Volatility: Meaning in Finance https://www.investopedia.com/terms/v/volatility.asp*

20) ***Algorithmic :*** *Geeksforgeeks webpage (n.d.) : What is Algorithm | Introduction to Algorithms https://www.geeksforgeeks.org/introduction-to-algorithms/*

21) ***web development***: *MDN Web Docs. (n.d.).* ***Learn web development****. Mozilla Developer Network https://developer.mozilla.org/en-US/docs/Learn_web_development*

22) ***Graphical Data Visualization:*** *Geeksforgeeks webpage (n.d.). Data Visualization:https://www.geeksforgeeks.org/charts-and-graphs-for-data-visualization/*

23) ***AI/ML :*** *Google cloud webpage (n.d.). Artificial Intelligence (AI) Vs. Machine Learning (ML): How Do They Differ. https://cloud.google.com/learn/artificial-intelligence-vsmachine-learning?hl=en*