**Project Report**

****

Prepared by-

|  |  |  |
| --- | --- | --- |
| **Specialization** | **SAP ID** | **Name** |
| CSE-CCVT | 500077078 | Devang Tyagi |
| CSE-CCVT | 500076068 | Gaurav Kumar |
| CSE-CCVT | 500076054 | Vaibhavi Shukla |
| CSE-CCVT | 500076112 | Vishesh Jain |

Mentor – **Mr. Amrendra Nath Tripathi**

Department Head (Dept. of Systematics) – **Mrs. Neelu Jyoti Ahuja**



Department of Systematics

School Of Computer Science

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,

DEHRADUN- 248007. Uttarakhand

**INTRODUCTION**

* In today’s world, you need to plan well and manage spending of your money. It’s very easy to let your spending get out of your hand if you don’t know how much of your money is disposable income.
* Everyone has many different types of expenditures in their life which they need to manage and spend properly according to their needs.
* ‘Smart Shopping Manager’ is such a type of program which can be used to optimize your budget in order to save your money.

**LITERATURE REVIEW**

* **Dominic Asamoah, Evans Baidoo, Stephen Opoku Oppong (2017). ‘Optimization memory using Knapsack Algorithm’, Department of Computer Science, I.J. Modern Education and Computer Science *(1****)*, in this paper, a Dynamic Programming algorithm is proposed for the 0/1 one dimensional Knapsack problem. Problem specific knowledge is integrated in the algorithm description and assessment of parameters with a specific end goal to investigate the execution of the finite-time implementation of dynamic programming.
* Dynamic Programming (DP) is an effective procedure that permits one to take care of a wide range of source of problems in time O(n2) or O(n3) for which innocent methodology would take exponential time.
* **Maya Hristakeva, Dipti Shreshta. ‘Different Approaches to Solve the 0/1 Knapsack Problem’, Computer Science Department, Simpson College, Indianola, IA 50125 *(2)***, in this paper, Dynamic Programming is defined as a technique for solving problem whose solutions satisfy recurrence, relations with overlapping subproblems. Dynamic Programming requires a two-dimensional array with rows equal to the number of items and columns equal to the capacity of the knapsack.
* In the implementation of the algorithm instead of using two separate arrays for the weight in the values of the items, we used one array Items of type item, where item is as structure with two fields weight and value.

**PROBLEM STATEMENT**

* Money is an important aspect of modern life. Managed well, you can save more and remain stable but if used poorly, results could be devastating. You need to optimize beforehand how to use your money to maximize efficiency.
* Without proper attention, it can lead to overspending which would ultimately affect your short-term and long-term goals negatively.
* But how to manage your money? That’s where ‘Smart Shopping Manager’ comes in.
* Using ‘Smart Shopping Manager’, you can shop accordingly without spending any reckless amount of money.

**OBJECTIVES**

The main objectives aimed to be achieved through this program are:

* To promote the use of optimization of budget before spending it.
* Save user’s money by helping them with shopping according to their needs.

**METHODOLGY**

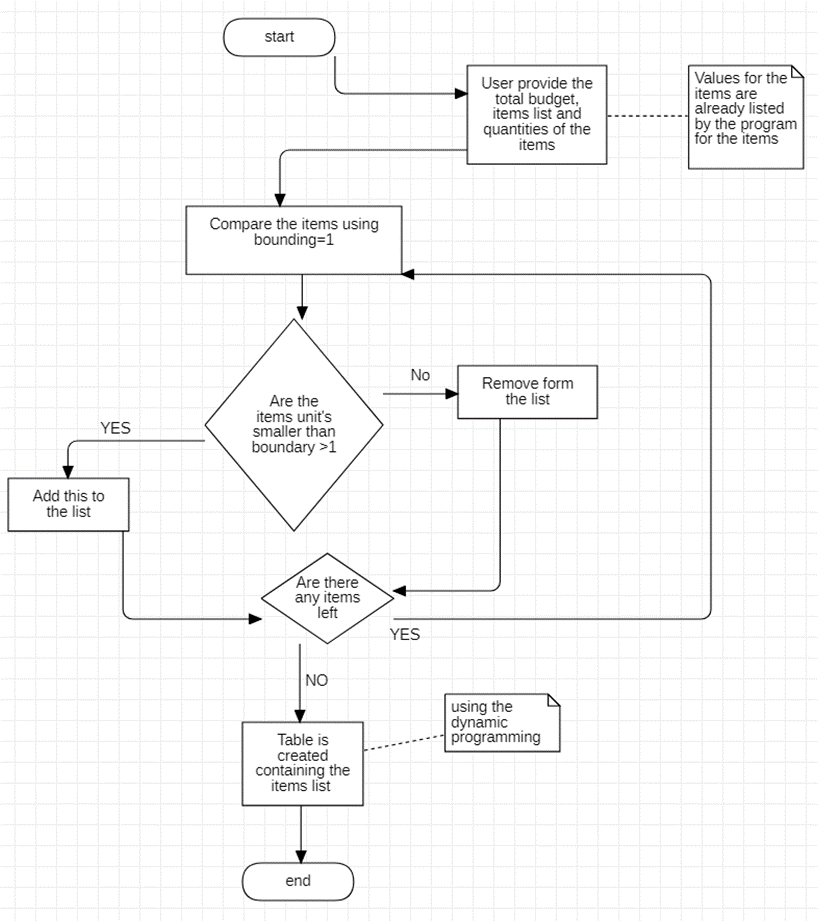
Here, we are using Agile Model and following are the steps:

1. **Planning:** The main requirement is that finances are needed to be managed to avoid excess expenditure. For money management, we are using Knapsack algorithm to optimize it and dynamic programming to tackle the Knapsack problem in an efficient manner.
2. **Designing:** Now, we need a program that can be used to reduce the reckless spending of money and better optimization of it.
3. **Development:** For the development of the program, Knapsack algorithm is used to optimize with Dynamic Programming in the program using Java.
4. **Testing:** And finally, during testing process, it gives desired output as a software which is able to calculate how to shop according to the needs of the user.

**EXPERIMENTAL SETUP**

* Hardware requirements - 64-bit operating system, x64-based processor  
   Processor - Intel® core™ i5-8250U CPU @ 1.60 GHz  
   RAM – 8 GB
* Software requirements - VirtualBox, ubuntu (version 20.0.04), open jdk (version 14)
* Minimum requirements on client/user End-   
   32-bit operating system, x32-based processor  
   Processor - Intel® core™ i5-8250U CPU @ 1.60 GHz 1.80 GHz  
   RAM – 4 GB

**FLOWCHART**



**Algorithmic Approach**

**0-1 Knapsack Algorithm:**

1. The 0/1 knapsack problem means that the items are either completely or no items are filled in a knapsack.  
   In the 0–1 Knapsack problem, we are given a set of items, each with a weight and a value, and we need to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
2. **Parameters**:

* Quantities of the items in terms of units, for example, 9 units.
* Items of the products’ name which user wants to buy.
* Values that decide the importance of the item.
* It picks up the item which does not exceed the default limit and the value of the items is maximized.

**References**

1. **Dominic Asamoah, Evans Baidoo, Stephen Opoku Oppong (2017).**‘Optimization memory using Knapsack Algorithm’  
   <https://www.mecs-press.org/ijmecs/ijmecs-v9-n5/IJMECS-V9-N5-5.pdf>
2. **Maya Hristakeva, Dipti Shreshta.**Different Approaches to Solve the 0/1 Knapsack Problem  
   <https://micsymposium.org/mics_2005/papers/paper102.pdf>
3. An Algorithm of 0-1 Knapsack Problem Based on Economic Model  
   <https://www.scirp.org/journal/paperinformation.aspx?paperid=39081>
4. A MAB-Based Knapsack Algorithm for Coexistence of LTE/WiFi in Resource Allocation Optimization [https://ieeexplore.ieee.org/abstract/document/9295891/metrics#metrics](https://ieeexplore.ieee.org/abstract/document/9295891/metrics)
5. A combination of the Knapsack algorithm and MIVES for choosing optimal temporary housing site locations: A case study in Tehran<https://www.sciencedirect.com/science/article/abs/pii/S2212420917303035>