Mid Term Report

on

**Smart Stock Portfolio Optimizer**

(0/1 Knapsack Problem Solver)

Project-I

**BACHELOR OF TECHNOLOGY**

(Computer Science and Engineering)



**SUBMITTED BY:**

Vikash Kumar, 2420367

Ishu Barman, 2420358

Dharmu Kumar, 2420353

Shubham Tandon, 2320207

March 2025

**Under the Guidance of**

Dr. Ishu Sharma

Associate Professor

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

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Contents** | **Page No** |
| 1. | Introduction | 1-3 |
| 2. | System Requirement | 3-4 |
| 3. | Software Requirement Analysis | 4-6 |
| 4. | Software Design | 6-22 |
| 5. | Implementation | 23 |
| 6. | References | 24 |

# Chapter 1.Introduction:

In today’s financial markets, investors have to choose from a variety of stocks, but not everyone has substantial funds to invest. They often have a limited amount of money, meaning they can only buy a few stocks that fit within their budget.

The goal of this project is to develop a tool that helps investors choose the best stocks within their budget. This tool will utilize the Knapsack Dynamic Problem approach, a common optimization technique. It involves selecting stocks within a limited budget to maximize value or returns while balancing risk-adjusted returns.

In this project, the objective is to create a tool that enables investors to select stocks such that the total cost does not exceed the given budget while maximizing the total expected return. The challenge is to identify the best stocks from a given list that fit within financial constraints while also offering the highest return based on risk-adjusted values. This tool aims to provide investors with an optimal strategy to make informed financial decisions and maximize profits.

## 1.1 Problem Context:

Stocks investments are risky, but they gives the potentials for returns. The main challenge is to select the right stock within the budget that gives the best return. To do this, each stock has a cost per share (similar to weight in the knapsack problem) and an expected return (similar to value in Knapsack problem).

**For Example :**

Stock A: Cost = Rs. 1500, Expected Return = Rs. 300

Stock B: Cost = Rs. 900, Expected Return = Rs. 100

Stock C: Cost = Rs 2000, Expected Return = Rs. 500

Stock D: Cost = Rs. 2500, Expected Return = Rs. 800

Budget = Rs 4300

Figure 1.1 : Graph showcasing Cost and Return of the above data

The optimal Selection for this would be to choose **Stock D** and **Stock A** for a total cost of Rs. 4300 and a **return of Rs. 1100.**

## 1.2 Problem Statement:

In the context of this Project, the objective is to develop a tool that allows an investor to select a stock in such a way that the total cost does not exceeds a given total budget, and at that same time total expected return is maximized.

## 1.3 The objectives of this project are:

* **To create a tool for optimizing stock portfolios** based on stock cost, expected return, and risk. The goal is to help investors choose the best combination of stocks that maximize their return while staying within their budget.
* **To use Dynamic Programming (DP)** to solve the problem of selecting the best portfolio. This approach will help in finding the best combination of stocks that gives highest return without exceeding the given budget.
* **To design an easy-to-use interface** for entering stock information (budget, value, return) The interface will simple and clear, allowing user to easily input their data and understand the results.
* **To clearly show the results of the optimization** with a summary of the best stock choices, total cost, and expected returns. This will make easy for the user to choose that which combination of stocks are best for their budget and goals.
* **To ensure that the tool can handle different stock list and budget sizes**. Weather the user has a few stocks or a large list of stocks, the tool will be flexible and work with different budgets to help make the best investment choices.
* **To consider the risk of each stock** and include that in portfolio section. This will help investors choose stocks that match their comfort with risk while still focusing for the best possible return.

## Chapter 2.System Requirement:

**2.1 Client-Side (User) Requirements:**

* **Web Browser**: Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions)
* **Internet Connection**: Stable connection for interacting with the server
* **Device**:
  + Laptop/Desktop (recommended)
  + Minimum screen resolution: 1280x720
* **OS Compatibility**: Cross-platform (Windows/Linux/macOS)

**2.2 Server-Side Requirements:**

* **Operating System**: Windows/Linux server
* **Web Server**: Apache/Nginx (if deploying manually)
* **Backend Language**: C++ (for Knapsack logic, compiled as CGI or integrated with backend stack if applicable)
* **Database (if applicable)**:
  + MongoDB / MySQL / PostgreSQL (if storing user data or stock info)
* **Storage**: 1 GB (minimum for hosting source code, assets, and output logs)
* **RAM**: Minimum 2 GB for local server testing; 4 GB+ for production

**2.3 Development Tools Used:**

* **Frontend**: HTML, CSS, JavaScript
* **Backend**: JavaScript (Knapsack algorithm logic)
* **IDE**: Visual Studio Code / Sublime Text / Any code editor
* **Version Control**: Git & GitHub (for collaboration and backup)

**2.4 Functional Requirements:**

* Allow users to input:
  + Stock names, costs, expected returns, and budget
* Display:
  + Maximum return possible
  + List of selected stocks
* Web interface should be responsive and mobile-friendly.

## Chapter 3.Software Requirement Analysis:

This Software Requirement Analysis defines the functionalities and constraints of the web-based **Stock Portfolio Optimization Tool**. The goal of the system is to help users select an optimal set of stocks within a specified budget using the **0/1 Knapsack algorithm**.

**3.1 Functional Requirements:**

* **User Input Interface:**
  + Allows users to enter stock names, their respective costs, expected returns, and a total investment budget.
* **Knapsack Optimization Logic:**
  + Applies the 0/1 Knapsack algorithm (written in JavaScrip) to determine the best combination of stocks to maximize return within the given budget.
* **Result Display**
  + Displays selected stocks along with the total cost and return in a clear, structured format on the webpage.
* **Input Validation:**
  + Ensures that users do not enter negative values or leave any required fields blank.
* **Reset Option:**
  + Users can reset or clear inputs using a reset button.

**3.2 Non-Functional Requirements:**

* **Performance:**
  + The system should generate results in less than 2 seconds for inputs up to 1000 stock items.
* **Usability:**
  + The interface is user-friendly and responsive, designed for ease of use even for non-technical users.
* **Portability:**
  + Works seamlessly across major web browsers (Chrome, Firefox, Edge) and platforms (Windows/macOS/Linux).
* **Maintainability:**
  + The code is modular and clean, making it easy to understand and update in the future.

**3.3 Assumptions and Constraints:**

* The project does not include user authentication or a login system.
* No external database is used; all input is taken directly from the user via the form.
* Stock data is manually entered by the user — real-time API integration is not included.
* The algorithm is integrated and executed in a manner that suits the static nature of the GitHub Pages deployment (e.g., via precompiled logic or simulated interaction).

**Chapter 4.Software Design:**

Software design describes the architecture and logical flow of the system. It serves as a blueprint for developers to implement the system’s components. This section outlines how the system is structured and how its components interact.

**4.1 System Architecture:**

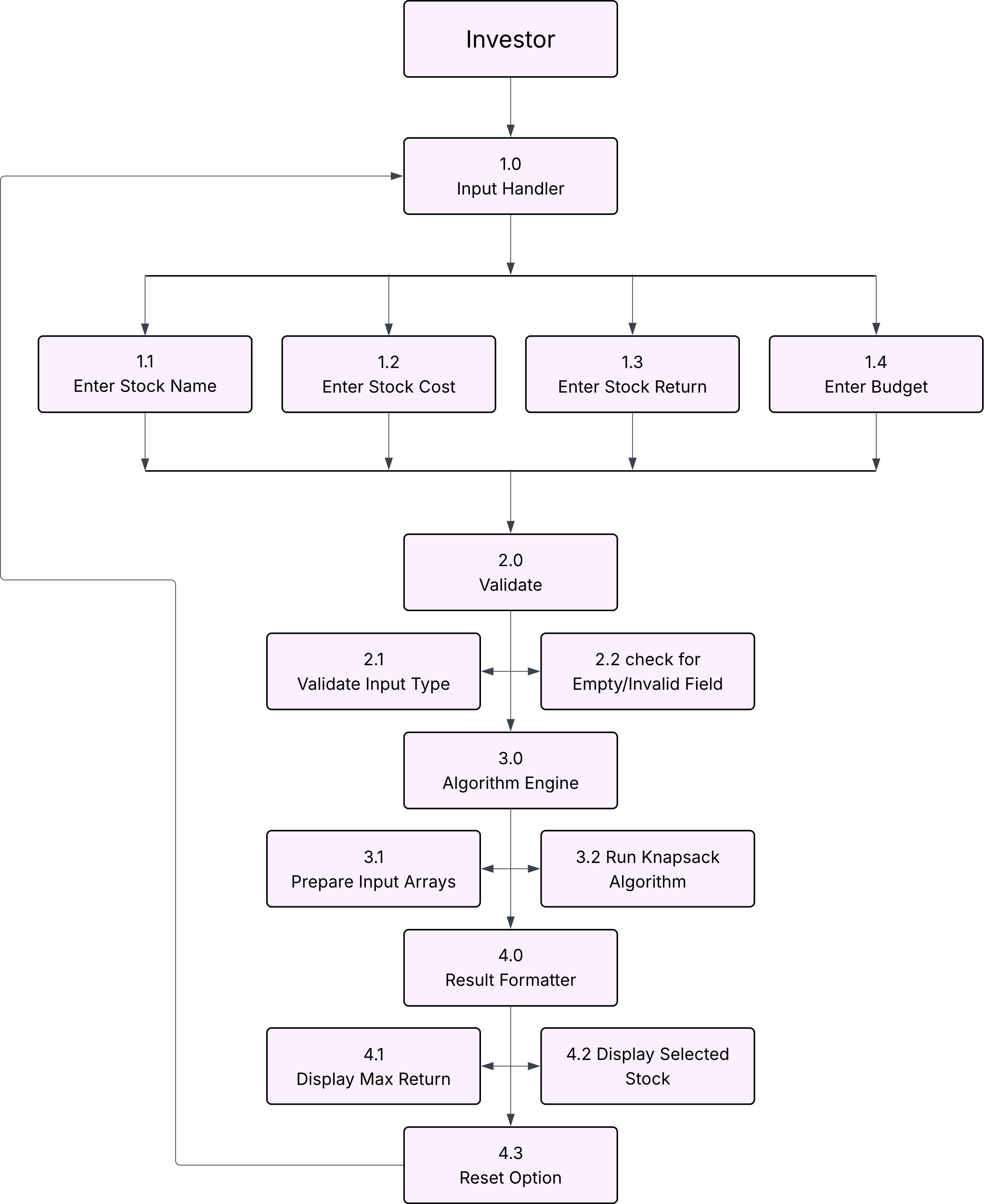
The system follows a **modular web-based architecture**, consisting of the following layers:

* **Presentation Layer (Frontend)**
  + Handles user interaction and input/output display
  + Built using HTML, CSS, and JavaScript
* **Logic Layer (Computation)**
  + Executes the 0/1 Knapsack algorithm using C++ logic
  + Processes input data and generates optimized results
* **Deployment Layer**
  + Hosted on GitHub Pages as a static website
  + Provides global access to the tool through a web browser

**4.2 Module Description:**

| **Module Name:** | **Description:** |
| --- | --- |
| **Input Module** | Collects user input: stock names, cost, returns, and budget |
| **Validation Module** | Checks that input is complete and valid |
| **Computation Module** | Applies the 0/1 Knapsack algorithm using C++ logic to calculate optimal stock selection |
| **Result Module** | Displays selected stocks, total cost, and total return on the screen |
| **Reset Module** | Allows users to reset the form and start fresh |

**4.3 Data Flow Diagram:**

****

**Chapter 5.Implementation:**

This section describes how the project was developed and how each module was implemented. It explains the process followed from designing the UI to integrating the knapsack logic and finally deploying the project online.

**5.1 Implementation Details:**

**5.1.1 Frontend Development**

* The user interface was developed using **HTML** and styled using **CSS**.
* **JavaScript** was used to manage user interactions, validate inputs, and display the output dynamically.
* A simple and responsive layout was created to ensure good usability across devices.

**HTML Code (index.html)**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Stock Portfolio Optimization (0/1Knapsack Problem)</title>

  <link rel="stylesheet" href="styles.css">

</head>

<body>

  <div class="area">

    <ul class="circles">

      <li></li>

      <li></li>

      <li></li>

      <li></li>

      <li></li>

      <li></li>

      <li></li>

      <li></li>

      <li></li>

      <li></li>

    </ul>

    <div class="container">

      <div class="heading">

        <h1>Smart Stock Portfolio Optimization</h1>

      </div>

      <!-- Stock Name and Stock Cost Input -->

      <div class="input-box">

        <div class="input-box1">

          <div class="input-group1">

            <!-- <label for="stock-name">Stock Name:</label> -->

            <input type="text" id="stock-name" placeholder="Enter Stock Name">

          </div>

          <!-- Stock Return Input -->

          <div class="input-group2">

            <label for="stock-cost">Stock Cost:</label>

            <input type="number" id="stock-cost" placeholder="Enter the Cost of the Stock">

            <label for="stock-return">Stock Return:</label>

            <input type="number" id="stock-return" placeholder="Enter the Return of the Stock">

            <!-- <label for="stock-return">Expected Return:</label>

        <input type="number" id="stock-return" placeholder="Enter the Return Value of the Stock"> -->

            <!-- Add stock Button -->

            <div class="add-stock">

              <button onclick="addStock()">Add Stock</button>

            </div>

          </div>

        </div>

        <!-- Stock List -->

        <div class="result-box">

          <h3>Stock List:</h3>

          <table id="stock-list">

            <thead>

              <tr>

                <th>Name</th>

                <th>Cost</th>

                <th>Return</th>

              </tr>

            </thead>

            <tbody></tbody>

          </table>

          <!-- Budget Input -->

          <div class="input-group3">

            <!-- <label for="budget">Enter Budget:</label> -->

            <input type="number" id="budget" placeholder="Enter Budget">

          </div>

          <!-- Calculate Button -->

          <div class="calculate">

            <button onclick="calculatePortfolio()">Calculate Portfolio</button>

          </div>

        </div>

        <!-- Result -->

        <div id="result"></div>

      </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

**CSS Code (style.css)**

body {

  font-family: Arial, sans-serif;

  background-color: #441752;

  margin: 0;

  padding: 20px;

}

.container {

  position: relative;

  max-width: 800px;

  max-height: 800px;

  margin: 0 auto;

  background-color: #A888B5;

  padding: 20px;

  border-radius: 18px;

  box-shadow: 0 0 10px rgba(255, 17, 17, 0.1);

}

.heading {

  margin-left: 145px;

}

h1 {

  text-align: center;

}

.input-group1 {

  margin-bottom: 15px;

  align-content: center;

  margin-left: 250px;

}

.input-box1 {

  background-color: #6b569c;

  border-radius: 10px;

  padding: 20px;

}

.input-group3 {

  /\* margin-bottom: 15px; \*/

  align-content: center;

  margin-left: 250px;

  margin-top: 15px;

}

input {

  padding: 10px;

  margin-right: 10px;

  margin-bottom: 10px;

  width: 200px;

}

button {

  padding: 10px 20px;

  background-color: #28a745;

  color: white;

  border: none;

  cursor: pointer;

  border-radius: 5px;

}

button:hover {

  background-color: #a2552f;

}

table {

  width: 100%;

  border-collapse: collapse;

  margin-top: 20px;

}

th,

td {

  padding: 10px;

  text-align: center;

  border: 1px solid #ddd;

}

#result {

  margin-top: 20px;

  padding: 15px;

  background-color: #f8d7da;

  color: #721c24;

  border: 1px solid #f5c6cb;

}

.add-stock {

  /\* margin-top: 20px;

  padding: 15px;

  background-color: #d4edda;

  color: #595959;

  border: 1px solid #c3e6cb; \*/

  margin-left: 310px;

  margin-bottom: 15px;

}

.calculate {

  margin-left: 290px;

  margin-bottom: 15px;

}

.input-box {

  background-color: #6b569c;

  border-radius: 10px;

  padding: 20px;

}

.result-box {

  background-color: #8174A0;

  border-radius: 10px;

  padding: 20px;

}

h1 {

  text-align: center;

  width: 30ch;

  font-family: monospace;

  font-family: cursive;

  text-wrap: nowrap;

  overflow: hidden;

  animation: typing 3s steps(40) infinite alternate-reverse;

}

@keyframes typing {

  from {

    width: 0ch;

  }

}

.context {

  width: 100%;

  position: absolute;

  top: 50vh;

}

.area {

  background: #ffffff00;

  width: 100%;

  height: 100vh;

  position: absolute;

}

.circles {

  position: absolute;

  top: 0;

  left: 0;

  width: 100%;

  height: 100%;

  overflow: hidden;

}

.circles li {

  position: absolute;

  display: block;

  list-style: none;

  width: 20px;

  height: 20px;

  background: rgba(115, 113, 21, 0.454);

  animation: animate 25s linear infinite;

  bottom: -150px;

}

.circles li:nth-child(1) {

  left: 25%;

  width: 80px;

  height: 80px;

  animation-delay: 0s;

}

.circles li:nth-child(2) {

  left: 10%;

  width: 20px;

  height: 20px;

  animation-delay: 2s;

  animation-duration: 12s;

}

.circles li:nth-child(3) {

  left: 70%;

  width: 20px;

  height: 20px;

  animation-delay: 4s;

}

.circles li:nth-child(4) {

  left: 40%;

  width: 60px;

  height: 60px;

  animation-delay: 0s;

  animation-duration: 18s;

}

.circles li:nth-child(5) {

  left: 65%;

  width: 20px;

  height: 20px;

  animation-delay: 0s;

}

.circles li:nth-child(6) {

  left: 75%;

  width: 110px;

  height: 110px;

  animation-delay: 3s;

}

.circles li:nth-child(7) {

  left: 35%;

  width: 150px;

  height: 150px;

  animation-delay: 7s;

}

.circles li:nth-child(8) {

  left: 50%;

  width: 25px;

  height: 25px;

  animation-delay: 15s;

  animation-duration: 45s;

}

.circles li:nth-child(9) {

  left: 20%;

  width: 15px;

  height: 15px;

  animation-delay: 2s;

  animation-duration: 35s;

}

.circles li:nth-child(10) {

  left: 85%;

  width: 150px;

  height: 150px;

  animation-delay: 0s;

  animation-duration: 11s;

}

@keyframes animate {

  0% {

    transform: translateY(0) rotate(0deg);

    opacity: 1;

    border-radius: 0;

  }

  100% {

    transform: translateY(-1000px) rotate(720deg);

    opacity: 0;

    border-radius: 60%;

  }

}

**Js (javascript) Code (script.js)**

let stocks = [];

function addStock() {

  // Get stock details

  const name = document.getElementById("stock-name").value;

  const cost = parseInt(document.getElementById("stock-cost").value);

  const returnVal = parseInt(document.getElementById("stock-return").value);

  if (!name || isNaN(cost) || isNaN(returnVal)) {

    alert("Please enter valid stock details.");

    return;

  }

  // Add to stocks list

  stocks.push({ name, cost, return: returnVal });

  // Update stock list table

  updateStockList();

  // Clear input fields

  document.getElementById("stock-name").value = '';

  document.getElementById("stock-cost").value = '';

  document.getElementById("stock-return").value = '';

}

function updateStockList() {

  const stockTableBody = document.getElementById("stock-list").getElementsByTagName("tbody")[0];

  stockTableBody.innerHTML = '';

  stocks.forEach(stock => {

    const row = document.createElement("tr");

    row.innerHTML = `<td>${stock.name}</td><td>${stock.cost}</td><td>${stock.return}</td>`;

    stockTableBody.appendChild(row);

  });

}

function calculatePortfolio() {

  const budget = parseInt(document.getElementById("budget").value);

  if (isNaN(budget) || budget <= 0) {

    alert("Please enter a valid budget.");

    return;

  }

  // Get stock data (cost and return values)

  const costs = stocks.map(stock => stock.cost);

  const returns = stocks.map(stock => stock.return);

  const n = stocks.length;

  // Knapsack dynamic programming solution

  const result = knapsack(budget, costs, returns, n);

  // Display the result

  const selectedStocks = result.selectedItems.map(i => stocks[i].name).join(", ");

  const maxReturn = result.maxValue;

  document.getElementById("result").innerHTML = `

    <strong>Max Return: </strong>${maxReturn}<br>

    <strong>Selected Stocks: </strong>${selectedStocks}

  `;

}

function knapsack(capacity, costs, returns, n) {

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

  // Dynamic programming approach to solve the Knapsack problem

  for (let i = 1; i <= n; i++) {

    for (let w = 0; w <= capacity; w++) {

      if (costs[i - 1] <= w) {

        dp[i][w] = Math.max(

          returns[i - 1] + dp[i - 1][w - costs[i - 1]],

          dp[i - 1][w]

        );

      } else {

        dp[i][w] = dp[i - 1][w];

      }

    }

  }

  // Find the selected items

  const selectedItems = [];

  let w = capacity;

  for (let i = n; i > 0; i--) {

    if (dp[i][w] !== dp[i - 1][w]) {

      selectedItems.push(i - 1);

      w -= costs[i - 1];

    }

  }

  return {

    maxValue: dp[n][capacity],

    selectedItems

  };

}

**5.1.2 Input Handling:**

* Users can input multiple stocks with their **names, costs, and expected returns**, along with a **budget**.
* Input fields are validated using JavaScript to avoid empty or invalid entries.

**5.1.3 Algorithm Integration:**

* The **0/1 Knapsack Algorithm** was implemented in **C++** to compute the optimal stock selection.
* The logic is integrated in a way that suits a static web deployment (e.g., compiled or simulated logic).

**5.1.4 Output Display:**

* After processing, the selected stocks, their total cost, and total return are displayed in a clean format.
* Users can reset the form to start a new calculation.

**5.2 Deployment:**

* The project was deployed on **GitHub Pages**, which allows hosting static websites directly from a GitHub repository.
* The final version is accessible publicly via a shareable link and works across major browsers.

**7.References:**

**Web Technologies:**

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

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

**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>

**Development Environment & Tool:**

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

**Version Control and Collaboration:**

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

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

**Others:**

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