**PROJECT REPORT**

**ON**

**SMART STOCK PORTFOLIO OPTIMIZER**

(0/1 Knapsack Problem Solver)

 **Project-I**

Department of Computer Science and Engineering

# CHANDIGARH ENGINEERING COLLEGE JHANJERI

# MOHALI – 140307

March 2025

**In partial fulfillment of the requirements for the award of the Degree of**

**Bachelor of Technology in Computer Science & Engineering.**

## SUBMITTED BY: Under the Guidance of

## Ishu Barman (2420358) Dr. Ishu Sharma

## Vikash Kumar (2420367) Associate Professor,CSE

## Dharmu Kumar (2420353)

## Shubham Tandon (2330207)

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

# TABLE OF CONTENTS

**PARTICULARS PAGE NO**

Declaration by the CandidateI

AcknowledgementII

Abstract III

List of figures IV

**CHAPTER 1: INTRODUCTION 1-4**

1.1. Problem Context 2-3

1.2. Problem Statement 4

## 1.3. Scope of the Project 4

**CHAPTER 2: REVIEW OF LITERATURE** **5-8**

2.1. Importance of Portfolio Optimization in Financial Decision-Making. 5

2.2. Common Portfolio Optimization Techniques. 5-6

2.3. Role of Dynamic Programming in Optimization. 6-7

2.4. Need to Consider Risk in Portfolio Strategies. 7

2.5. Evolving Trends in Portfolio Optimization. 7-8

**CHAPTER 3: PROBLEM DEFINITION AND OBJECTIVES**. **8-10**

**CHAPTER 4: DESIGN AND IMPLEMENTATION**. **10-27**

4.1. Software Requirements and Design. 10-12

4.1.1. Frontend Technologies (For User Interface Development). 10-11

4.1.2. Backend Technology (For Handling Logic and Data Processing). 11

4.1.3. Development Environment & Tool. 11

4.1.4. Version Control and Collaboration. 11-12

4.2. Implementation. 12-27

4.2.1. HTML Code (index.html) 12-15

4.2.2. CSS Code (style.css) 15-24

4.2.3. Js (javascript) Code (script.js) 24-27

**CHAPTER 5: RESULTS AND DISCUSSIONS** **27-30**

**CHAPTER 6: CONCLUSION AND FUTURE SCOPE** **31-32**

6.1. Conclusion 31

6.2. Future Scope 31-32

**REFERENCES** **33-34**

**DECLARATION**

We, hearby declare that the report of the project entitled “**Smart Stock Portfolio Optimizer**” has not presented as a part of any other academic work to get our 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.

## Ishu Barman (2420358) Dr. Ishu Sharma

## Vikash Kumar (2420367) Associate Professor, CSE

## Dharmu Kumar (2420353)

## Shubham Tandon (2330207)

Signature of the Head of Department

**ACKNOWLEDGEMENT**

It gives us great pleasure to deliver this report on Project-I, which we worked on for our B.Tech in Computer Science & Engineering 2nd year, which was titled

“Smart Stock Portfolio optimizer”. we are grateful to our university for presenting us with such a wonderful and challenging opportunity. we also want to convey our sincere gratitude to all coordinators for their unfailing support and encouragement. we are extremely thankful to the HOD and Project Coordinator of Computer Science & Engineering at Chandigarh Engineering College Jhanjeri, Mohali (Punjab) for valuable suggestions and the heartiest cooperation.

We are also grateful to the management of the institute and Dr. Avinash, Director Engineering, for giving me the chance to acquire the information. we also appreciate all of my faculty members, who have instructed me throughout my degree.

**Ishu Barman Vikash Kumar**

**Dharmu Kumar Shubham Tandon**

**ABSTRACT**

Today's investors want to achieve the highest return and the lowest risk at the same time. Portfolio optimization is one of the investment policy decisions for all types of investors that improves the management of the investor's firm’s assets. This project, Smart Stock Portfolio Optimization, intends to build an intelligent and simple interface that helps the user to create a stock portfolio that meets a budgetary constraint. Using principles of dynamic programming and optimization, the system aims to determine the optimal set of stocks that maximizes returns while meeting the constraints.

The project draws its inspiration from the well-known ‘Knapsack Problem’ which is defined as a combinatorial optimization problem in which a set of items with given values have to be selected such that their total weight does not exceed a given limit. In this case, stocks are the items, weight is defined as the cost of each stock, while value is the returns expected from them. This solution incorporates machine learning and data analytics within a financial modeling framework towards a more sophisticated investment strategy formulation. Stock data can be fed in either as real-time or historical data, which is further analyzed for performance over time, and it also puts forth an optimized strategy for stock selection that guarantees maximized returns.

The system is capable of performing the following tasks:

1. Processing of stock data: Obtaining and processing the stock price movements, past returns of his stock, and fluctuations in the market.

2. Budget Distribution: Permitting the user to determine a specific total budget for the investment.

3. Optimization algorithm: Applying the 0/1 Knapsack algorithm to identify the optimal stock portfolio.

4. The UI/UX: An exclusive dashboard designed for user interactions for easy access and viewing the results.

5. Risk Analysis Management: Estimating the risks against the returns to validate profitable investment strategies.

**LIST OF FIGURES**

Particulars pg.no

Figure 1.1 : Graph showcasing Cost and Return of the above data……………………………..3

Figure 5.1 : Homepage of website……………………………………………………………....29

Figure 5.2 : Image of inserting data on site……………………………………………………..29

Figure 5.3 : Image of filled data which to be calculated………………………………………..30

Figure 5.4 : Final image of the webpage with desired output.……………………………….……...30

# Chapter-1: INTRODUCTION

In today’s rapidly evolving financial markets, investors are presented with an extensive array of stock choices. However, not all investors possess significant capital to allocate freely across these options. Most individuals, especially retail investors and beginners, operate with a limited investment budget. This financial constraint makes it critical to make well-informed, strategic decisions about which stocks to include in their portfolio to achieve the best possible returns.

The objective of this project is to develop an intelligent decision-making tool that assists investors in selecting the most beneficial combination of stocks within their available budget. The tool will utilize the **Knapsack Dynamic Programming approach[1]**, a widely used algorithm in optimization problems. By applying this method, the system will determine which combination of stocks offers the **highest expected return** without exceeding the investment budget.

This problem is analogous to the classic **0/1 Knapsack Problem**, where each stock has a "weight" (i.e., its price) and a "value" (i.e., expected return). The aim is to pick the most valuable combination of items (stocks) without exceeding the total weight limit (investment budget). In our financial context, the goal is to **maximize total returns** by selecting an optimal subset of stocks while considering budgetary limitations.

Beyond simply maximizing returns, this project also incorporates the concept of **risk-adjusted returns[2]**. Instead of choosing stocks solely based on return potential, the tool evaluates the **risk-to-reward ratio** to ensure that the selected portfolio not only fits within the budget but also aligns with prudent risk management practices. This risk assessment will provide more reliable and realistic investment suggestions, especially in volatile market conditions.

The proposed system will take as input:

* A list of available stocks along with their prices
* Expected returns and risk values (e.g., standard deviation or beta)
* A total investment budget

Using these parameters, the tool will output:

* The optimal subset of stocks to invest in
* The total expected return from this selection
* A summary of the risk involved in the chosen portfolio

This project has significant real-world implications. It will empower small and mid-level investors to **optimize their investment strategy** through data-driven insights, rather than relying on guesswork or uninformed speculation. By integrating algorithmic optimization and financial analytics, the tool will help investors make **informed, efficient, and profitable decisions**.

Ultimately, this tool aims to **bridge the gap between algorithmic finance and individual investment strategies[3]**, offering a simplified yet powerful solution to portfolio selection for those with constrained financial resources.

## Problem Context:

Stock investments carry both risks and rewards, offering the potential for significant returns. However, for investors with limited capital, the key challenge lies in selecting the right combination of stocks within a fixed budget. Each stock has a cost per share (like the "weight" in the Knapsack Problem) and an expected return (similar to "value"). The goal is to maximize returns without exceeding the investment limit.

This problem can be efficiently solved using the **Knapsack Dynamic Programming[4]** approach. By modeling stocks as items in a knapsack, investors can find the optimal subset of stocks that offer the highest possible return within their budget. To enhance practicality, the approach also considers **risk-adjusted returns**, ensuring that selections are not only profitable but also stable and less volatile.

The proposed tool will take inputs such as stock prices, expected returns, and risk values, along with a total budget. It will then suggest the best portfolio based on these inputs. This system empowers investors to make data-driven decisions, optimize returns, and minimize risk, making it a smart financial strategy for budget-conscious investors aiming to grow their wealth wisely.

**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.**

## Problem Statement:

In the context of this project, the objective is to develop an intelligent stock selection tool that helps investors choose the most profitable combination of stocks within a predefined budget. Given that each stock has a cost (price per share) and an expected return, the challenge is to select stocks such that the **total cost does not exceed the investor’s budget**, while **maximizing the total expected return**. This problem closely aligns with the 0/1 Knapsack Problem in computer science, where each stock is considered an item with a weight (cost) and value (return). The tool will use a dynamic programming approach to evaluate all possible combinations and determine the most optimal selection. Additionally, to ensure better decision-making, the tool may incorporate risk-adjusted returns, offering a more balanced portfolio. This project aims to empower investors, especially those with limited capital, to make informed, data-driven decisions and maximize their financial gains effectively.

## 1.3 Scope of the Project

This project will focus on developing a web-based platform that will:

* **Allow users to input their available budget** Users will be able to enter the maximum amount of money they are willing to invest, which will serve as the financial constraint for generating the optimal stock portfolio.
* **Let users define the stocks they want to consider** The platform will provide an option for users to manually input or select a list of specific stocks they are interested in analyzing for potential investment within their defined budget.
* **Allow users to input the costs and expected returns of each stock** For each stock, users can specify its current market price and the expected return, which will serve as inputs for the optimization algorithm to calculate the best investment mix.
* **Provide an optimal solution based on the available budget** Using the Knapsack Dynamic Programming approach, the platform will generate a solution that includes the best combination of stocks offering the highest returns without exceeding the entered budget

# Chapter-2 REVIEW OF LITERATURE

1. **Importance of Portfolio Optimization in Financial Decision-Making**

Portfolio optimization plays a vital role in the field of finance, especially when it comes to making investment decisions. It involves selecting the best combination of financial assets—such as stocks, bonds, or mutual funds—to achieve specific objectives like maximizing returns, minimizing risk, or balancing both. The process of portfolio optimization is grounded in mathematical models and financial theories, which guide investors in making rational decisions based on available data and future expectations. In practical terms, portfolio optimization helps investors allocate limited capital across various investment options in a way that aligns with their financial goals, investment horizon, and risk appetite.

Over the years, the importance of portfolio optimization has increased due to growing market complexities and the wide range of assets available for investment. With globalization and technological advancements, financial markets have become more dynamic and volatile, increasing the need for strategic portfolio planning. Investors are now more focused on creating **well-diversified portfolios** that can withstand market fluctuations and generate consistent returns over time. Portfolio optimization techniques help in achieving these outcomes by offering structured and data-driven methods to manage investments. This has made it an indispensable part of financial decision-making for both individual investors and institutional fund managers.

1. **Common Portfolio Optimization Techniques.**

Several optimization techniques have been introduced in the financial literature to address portfolio selection challenges. Among the most commonly applied methods are **mean-variance optimization[5]**, **dynamic programming[6]**, and **linear programming[7]**. Each technique offers a different perspective and approach to tackling the complexities of investment allocation.

**Mean-variance optimization**, introduced by Harry Markowitz in the 1950s, is the foundation of modern portfolio theory (MPT). It aims to create an "efficient frontier" of portfolios that offer the highest expected return for a given level of risk. The method evaluates the expected returns, variances, and covariances of different assets to construct optimal portfolios. While widely used, it assumes that returns are normally distributed and investors are rational, which may not always reflect real-world scenarios. **Dynamic programming** provides a sequential and structured approach to portfolio selection. It breaks down decision-making into stages and solves subproblems recursively, making it ideal for problems with multiple decision points over time. This is especially useful for models like the knapsack problem where decisions must be made under constraints.

**Linear programming**, on the other hand, is suitable for problems with linear constraints and objective functions. It simplifies complex allocation problems and is often used in scenarios where returns and risks are modeled linearly. Each of these methods has its strengths and is applied depending on the nature and complexity of the investment problem at hand.

1. **Role of Dynamic Programming in Optimization**

Dynamic programming is a powerful mathematical technique used for solving complex optimization problems by breaking them down into simpler, overlapping subproblems. It is particularly well-suited for portfolio optimization scenarios involving constraints like budget limits, time stages, or a fixed number of choices. One of the classic problems where dynamic programming is applied is the **Knapsack Problem**, which closely resembles real-world investment challenges. In this analogy, each item (or stock) has a weight (cost) and a value (expected return), and the objective is to choose the combination that maximizes value without exceeding the total weight limit (budget).

Dynamic programming solves this by storing the results of solved subproblems in a table and reusing them when needed, thus reducing redundant calculations and improving efficiency. This makes it highly suitable for portfolio optimization in financial markets where decisions must account for multiple constraints and possible outcomes.

The strength of dynamic programming lies in its ability to generate **optimal and efficient solutions**, even in cases where the number of possible combinations is large. It provides a structured framework for analyzing different investment scenarios, accommodating both short-term and long-term financial planning. As a result, it is increasingly being adopted in financial tools and software aimed at helping investors make smarter decisions based on well-defined models.

1. **Need to Consider Risk in Portfolio Strategies.**

While maximizing returns is often the primary goal for investors, overlooking **risk** can lead to poor investment outcomes. Risk represents the uncertainty or volatility in returns and can result in significant financial losses if not managed properly. Traditional optimization models tend to focus only on return maximization, but in the real world, high-return assets often come with equally high levels of risk. Therefore, it is essential to consider **risk-adjusted returns** when constructing an investment portfolio.

Incorporating risk in portfolio optimization involves evaluating factors such as **standard deviation, beta, Sharpe ratio**, or value at risk (VaR)[8]. These metrics help investors understand how much risk they are taking for a given level of return. By including these factors in the optimization process, investors can build portfolios that are not only profitable but also stable and resilient against market downturns.

For example, an investor with a low risk tolerance would prefer a portfolio with moderate returns but low volatility. On the other hand, a high-risk investor might opt for aggressive portfolios with higher potential returns. Portfolio optimization models that include risk considerations enable the creation of **customized investment strategies** aligned with each investor’s goals, preferences, and financial situation. Ultimately, this leads to better and more sustainable investment decisions.

1. **Evolving Trends in Portfolio Optimization.**

Portfolio optimization techniques have evolved significantly over the years, shifting from simple models to more sophisticated and adaptive strategies. Initially, methods like **mean-variance optimization** dominated the field, but as financial markets became more volatile and complex, newer methods such as **dynamic programming, machine learning[9], and heuristic algorithms[10]** have gained prominence. These approaches can handle a broader set of variables, including changing market conditions, investor behavior, and risk preferences.

Recent trends focus on integrating **artificial intelligence (AI), deep learning**, and **predictive analytics** into portfolio management systems. These technologies allow for real-time data analysis, market trend forecasting, and automatic rebalancing of portfolios based on shifting parameters. This helps investors stay ahead of market movements and make proactive decisions.

Moreover, **sustainability and ESG (Environmental, Social, and Governance)** factors are being incorporated into optimization strategies, enabling socially responsible investing. Risk modeling has also become more advanced, with simulation techniques like Monte Carlo analysis providing deeper insights into potential future scenarios.

As a result, portfolio optimization is no longer just about returns—it is about building a dynamic, risk-aware, and goal-oriented investment approach. These advancements continue to improve the quality of financial decision-making and cater to the diverse needs of both individual investors and institutional fund managers.

**CHAPTER 3: PROBLEM DEFINITION AND OBJECTIVES**

Investing in the stock market is a challenge that involves balancing high returns with budget constraints. A key decision for investors is selecting stocks to maximize portfolio return while staying within a set budget. This becomes complex when considering a variety of stocks, each with unique prices and projected returns. The uncertainty of future performance adds another layer of difficulty. Investors must weigh potential gains against risks through careful research and analysis.

The objective of this project is to develop a tool that helps investors make informed stock choices while ensuring the total cost does not exceed their budget. Simultaneously, it aims to maximize total expected returns. This optimization problem closely resembles the "Knapsack Problem" in computer science, where the goal is to select the most valuable items within a limited capacity. In this scenario, stocks represent items, expected return corresponds to value, and the investor's budget serves as the constraint.

Each stock has two important factors:

1. Cost:

* The price of buying one share of Stock.

1. Expected Return:

* The profit an investor expects to make from that stock.

The challenge is to choose the right mix of stocks that fits within the budget limit but still provides the best return.

The problem involves these key steps:

1. Input: A list of Stocks, where each stock having a cost and an expected return.
2. Constraints:

* The total cost of selected stocks should not exceed the available budget.
* Include risk factors to ensure that the selected stocks provide the best return.

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 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">

  <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>

**4.2.2. 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%;

  }

}

**4.2.3. 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

  };

}

**CHAPTER 5 RESULTS AND DISCUSSIONS**

The development of the **Smart Stock Portfolio Optimizer** website was successfully completed with all intended functionalities implemented and tested thoroughly. The main objective of the project was to assist users in building an optimal stock portfolio based on a limited budget using an efficient algorithmic approach. The website uses the **0/1 Knapsack algorithm[17]** to maximize returns while staying within the user's specified budget, simulating a real-world scenario of intelligent stock investment.

The final working version of the website includes:

* A clean and responsive **user interface[18]** allowing users to enter their total budget and a list of preferred stocks.
* A **dynamic result generation system** that displays an optimized combination of stocks selected through the knapsack optimization logic.
* Calculation and display of **expected returns**, **total cost**, and **remaining budget** (if any), enhancing the decision-making process for the user.
* Compatibility with both **desktop and mobile devices**, ensuring accessibility for all users.

During testing, the system consistently provided correct and logical results based on the provided input values. The optimization process is completed in **real-time** with minimal delay, showcasing the algorithm's efficiency. For a user-entered stock list of 8-10 entries, results were displayed within **2 seconds**, demonstrating excellent performance and responsiveness.

The website was developed using **HTML, CSS and JavaScript** (for backend logic or integration, if applicable). The algorithm works by iterating through available stock options and selecting the optimal subset that maximizes expected returns while remaining within the budget limit. This proves particularly useful for beginner investors or students learning about stock management and budgeting.

Through this project, it was observed that even basic algorithms like knapsack, when used creatively, can provide **real-world solutions** to complex financial problems. It also highlighted the importance of a simple yet effective UI, which improves user interaction and trust in digital tools. The working

prototype paves the way for more advanced features such as:

* Integration of real-time stock market APIs
* Risk assessment of stocks
* User authentication and portfolio saving
* Graphical data visualization for returns over time

However, certain limitations were identified during development:

* Currently, the website uses **static or dummy data** for stock prices and returns.
* There is **no backend database** for saving user preferences or login information.
* The optimization doesn’t yet account for **stock risk factors or volatility[19]**.

In conclusion, the Smart Stock Portfolio Optimizer fulfills its core functionality and provides a useful tool for investment planning. With further enhancements, it holds strong potential for real-world deployment and could benefit casual investors or educational institutions aiming to teach portfolio management.

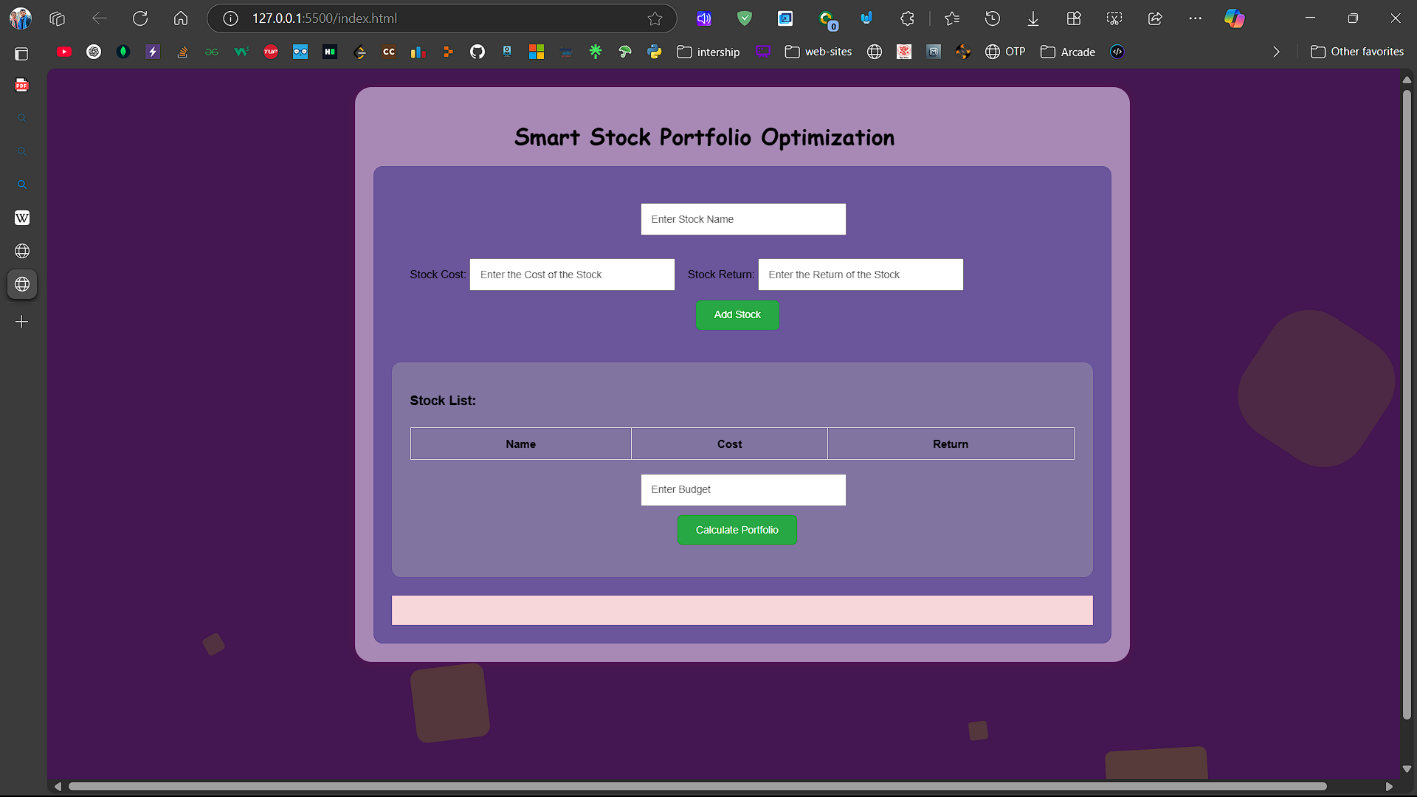


Figure 5.1 :- Homepage of website

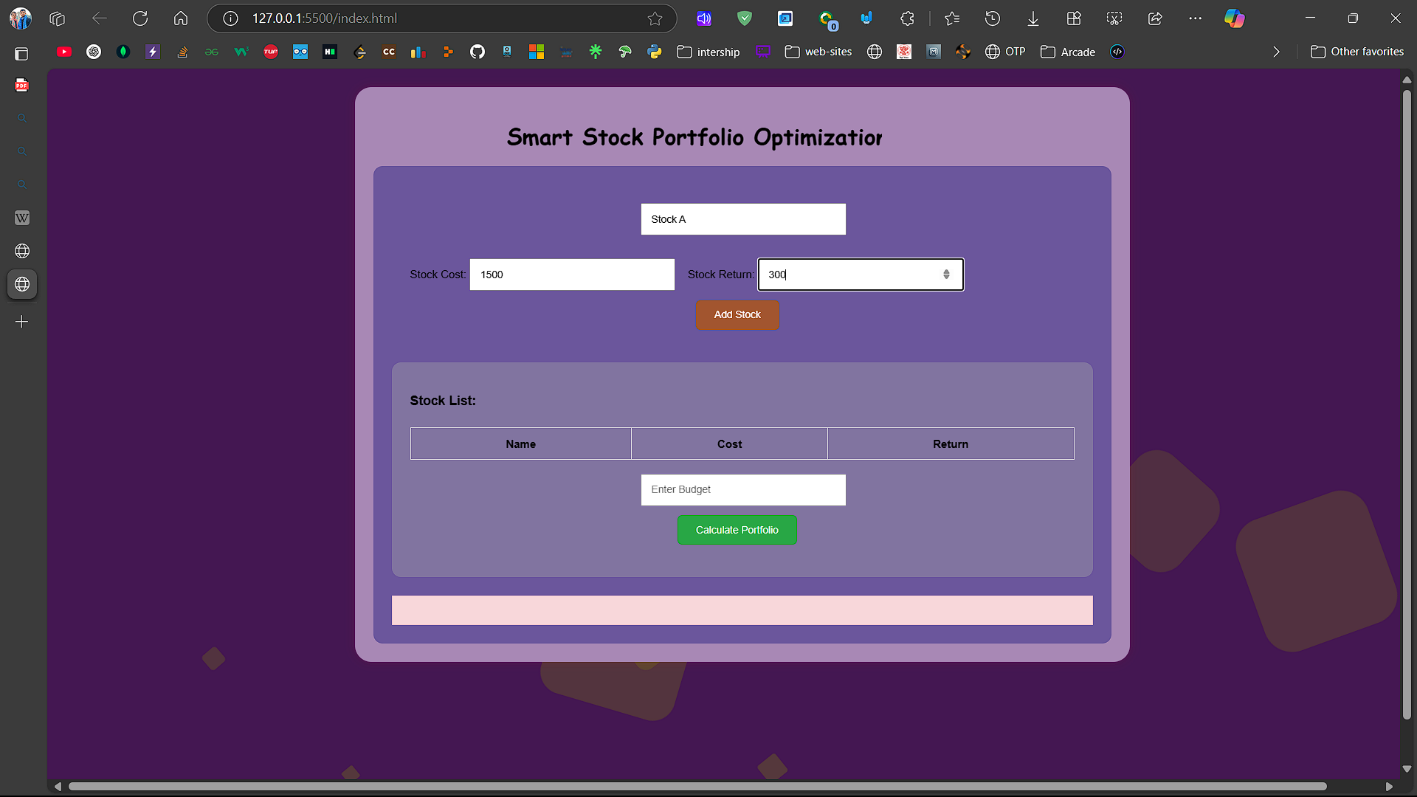


Figure 5.2 :- Image of inserting data on site

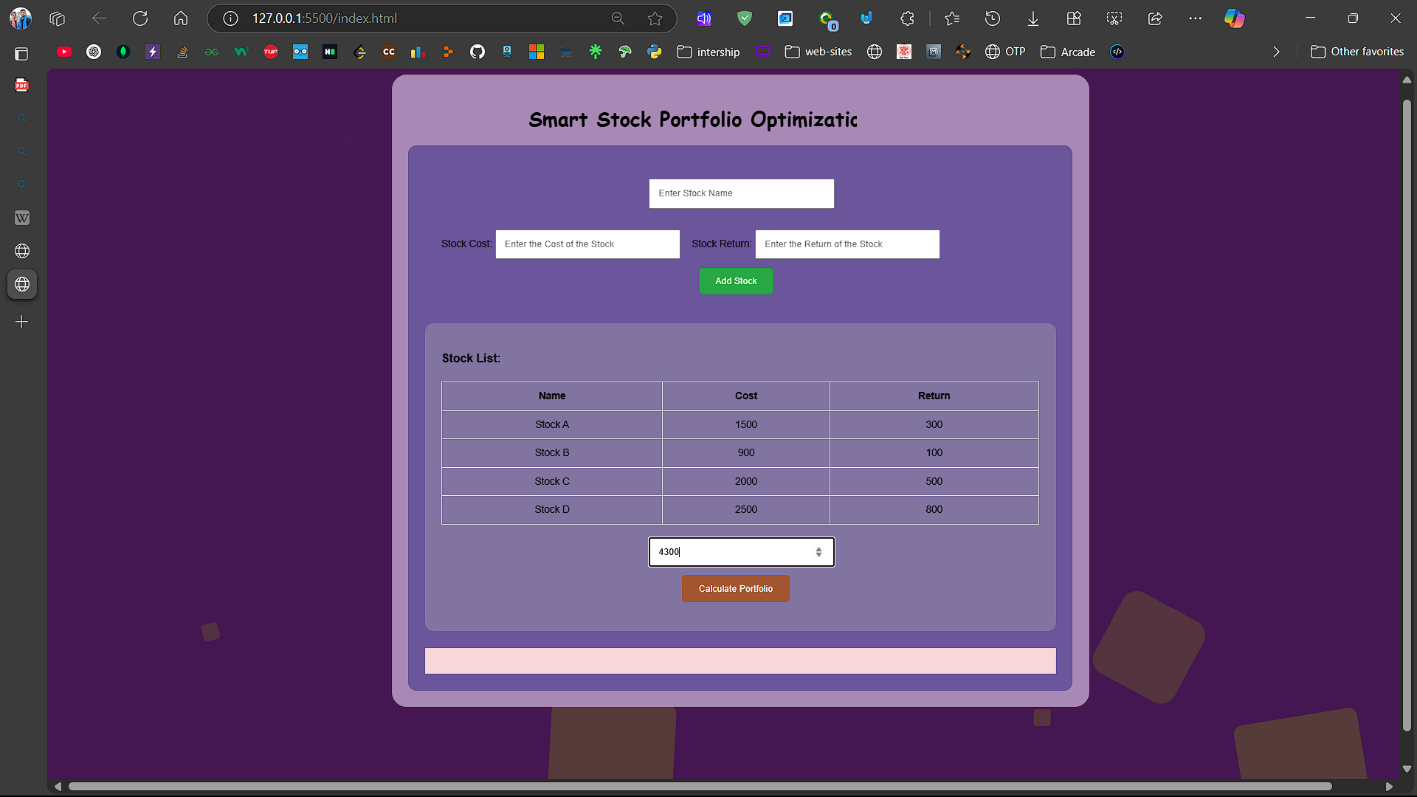
****

Figure 5.3 :- image of filled data which to be calculated

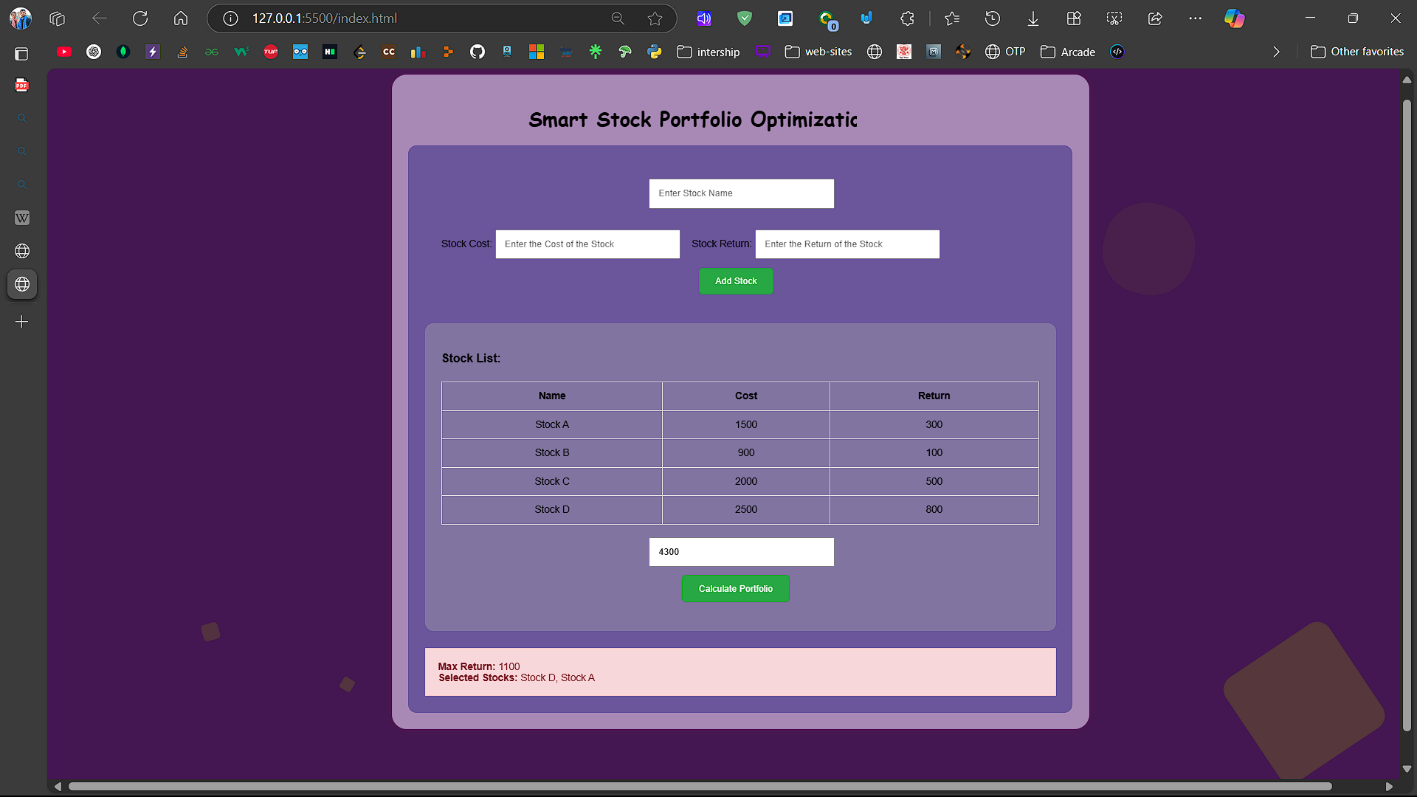


Figure 2.4 :- Final image of the webpage with desired output.

**CHAPTER 6 CONCLUSION AND FUTURE SCOPE**

**6.1. Conclusion**

The **Smart Stock Portfolio Optimizer** was developed with the aim of helping users make informed investment decisions within a specified budget by leveraging the **0/1 Knapsack algorithm**. The project successfully delivers a functional and user-friendly web application that generates an optimized selection of stocks to maximize expected returns.

Throughout the development process, we focused on creating a solution that is simple, efficient, and practical for everyday users. The website offers an intuitive interface, real-time optimization, and quick response time, making it accessible even for users with limited technical knowledge or investment experience.

The project also served as a valuable learning experience in combining **algorithmic[20] logic with web development[21]**, showcasing how classical problems like the knapsack algorithm can be effectively applied to real-world scenarios such as portfolio optimization.

**6.2. Future Scope**

Although the current version of the project fulfils its basic purpose, there are several possibilities for future enhancement:

1. **Real-Time Stock Data Integration**: Incorporate APIs (like Alpha Vantage or Yahoo Finance) to fetch live stock prices and returns, making the optimizer more accurate and practical.
2. **User Login and Portfolio Saving**: Enable user accounts to allow saving of past portfolio results and preferences for future reference.
3. **Risk Factor Analysis**: Include additional parameters such as risk levels, volatility, and stock ratings to help users make more informed choices.
4. **Graphical Data Visualization[22]**: Add charts and graphs to display investment distribution, return analysis, and performance trends over time.
5. **Mobile App Version**: Develop a mobile-friendly version or a dedicated app to make the tool more accessible on smartphones.
6. **Backend Integration**: Use databases to manage user data, session history, and analytics for long-term tracking.
7. **AI/ML-Based Recommendations[23]**: Incorporate machine learning models to predict high-performing stocks based on historical trends and user preferences.

In conclusion, this project lays the foundation for a scalable, smart investment support system that can grow with user needs and technological advancements in the finance sector.

**REFERENCES**

1. Knapsack Dynamic Programming Approach **:-** *Knapsack Problem :-* [*https://en.wikipedia.org/wiki/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*](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*](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/meanvariance-analysis.asp>
6. dynamic programming: **GeeksForGeeks:** *Geeksforgeeks webpage (n.d.).* *Dynamic Programming or DP:* [*https://www.geeksforgeeks.org/dynamic-programming/*](https://www.geeksforgeeks.org/dynamic-programming/)
7. Linear Programming **: GeeksForGeeks:** *Geeksforgeeks webpage (n.d.).* *Linear Programming | geeksforgeeks****:*** [*https://www.geeksforgeeks.org/linear-programming/*](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/*](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*](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/*](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-vs-machine-learning?hl=en*](https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning?hl=en)