OPTIMAL LENGTH UTILIZATION

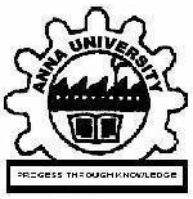
A MINI PROJECT REPORT

Submitted by

JAI SAARATHI R (221801019)
JOEL SAM O (221801021)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY:CHENNAI 600 025

NOVEMBER 2024

ANNA UNIVERSITY:CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Report titled "**OPTIMAL LENGTH UTILIZATION**" is the bonafide work of **JAI SAARATHI R (22180109), JOEL SAM O (221801021)** who carried out the work under my supervision.

SIGNATURE

Dr. J.M. Gnanasekar M.E., Ph.D.,
Professor and Head
Department of Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai – 602 105

SIGNATURE

Dr. P. Indira Priya M.E., Ph.D.,
Professor
Department of Artificial Intelligence and Data Science
Rajalakshmi Engineering College
Chennai – 602 105

Submitted for the project viva-voce examination held on……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.,** our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** and our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks **to Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. J.M. GNANASEKAR., M.E., Ph.D.,** Professor and Head of the Department, Department of Artificial Intelligence and Data Science for her guidance and encouragement throughout the project work. We are glad to express our sincere thanks and regards to our supervisor and coordinator, **Dr. P. INDIRA PRIYA., M.E., Ph.D.,** Professor, Department of Artificial Intelligence and Data Science for their valuable guidance throughout the course of the project.

Finally, we express our thanks for all teaching, non-teaching, faculty and our parents for helping us with the necessary guidance during the time of our project.

ABSTRACT

This project focuses on developing an optimized solution for pipe-cutting operations by utilizing algorithmic techniques to minimize waste and improve material usage efficiency. Pipe-cutting tasks in industrial settings often involve significant material wastage due to sub-optimal cutting patterns, which not only incurs higher costs but also contributes to environmental impact. Our solution addresses these challenges by implementing a smart pipe-cutting optimization system that can automatically determine the most efficient way to cut pipes based on input parameters, such as required cut lengths and available pipe dimensions.

The system leverages a Best Fit Decreasing (BFD) algorithm that generates cutting patterns aimed at maximizing resource utilization and minimizing waste. Through a user-friendly interface, operators can upload data, including required pipe dimensions and specific cut lengths. The system then calculates the optimal cutting patterns and provides a visual representation of pipe usage. Additional features include an analytics dashboard that reports performance metrics, such as total material waste, pipe utilization efficiency, and the number of pipes used. Moreover, a feedback module is integrated to enable continuous learning and improvement of the optimization process based on historical data.

Our approach ensures a cost-effective, sustainable, and highly automated pipe-cutting process, enabling industries to achieve their efficiency goals while reducing material costs and waste. The project contributes to advancing industrial efficiency practices by combining algorithmic rigor with practical visualization and data-driven insights, making it applicable in various fields that require precise material handling and waste reduced.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Efficient material utilization has become a pivotal goal in industrial manufacturing, where the rising costs of raw materials, environmental concerns, and the need for operational efficiency necessitate innovative solutions. The study at hand is focused on optimizing the process of pipe cutting, a common and resource-intensive task in industries that utilize pipes for various purposes. By examining ways to reduce waste, streamline operations, and enhance productivity, this study contributes to the broader objective of sustainable manufacturing practices.

1.1 GENERAL

Pipe cutting is a fundamental operation in industries like construction, plumbing, manufacturing, and automotive production, where pipes are custom-cut to meet specific length requirements. The challenge arises when standard pipe lengths, usually provided in bulk by suppliers, are cut down to required sizes, often resulting in leftover material that is wasted if it cannot be used efficiently. Such material losses accumulate significantly over time, impacting both the cost structure and the sustainability profile of manufacturing operations.

In recent years, advancements in data processing and optimization algorithms have presented new opportunities for tackling this issue. By developing an automated system to calculate the most efficient cutting patterns for pipes, industries can minimize waste and make better use of available materials. Optimization systems can help industries remain competitive by reducing costs, improving productivity, and contributing to a lower environmental footprint, aligning with the global push toward sustainable and responsible production practices.

1.2 NEED FOR THE STUDY

The importance of optimizing material usage in manufacturing processes extends beyond simply reducing costs. Several underlying motivations drive the need for this study:

- **Financial Efficiency**: Material costs are a substantial part of manufacturing expenses. Inefficient cutting can result in high waste, impacting the bottom line. An optimization solution can help industries reduce these costs by ensuring that raw materials are used in the most efficient manner possible.
- **Environmental Sustainability**: Industrial waste is a growing environmental concern, with legislative bodies and organizations worldwide urging companies to adopt more sustainable practices. By reducing waste through optimized cutting patterns, companies can significantly reduce their environmental impact and align with sustainability goals, ultimately contributing to responsible production and waste reduction.
- **Operational Productivity**: Manual planning for optimal cuts is not only time-consuming but also prone to human error. An automated system for generating cutting patterns can greatly increase productivity by saving time on planning and minimizing errors. This also frees up skilled labor to focus on other critical tasks in the manufacturing process.
- **Precision and Consistency**: An automated system can ensure that cutting patterns are consistent across production batches, ensuring that the same high standard of efficiency is applied every time. This consistency in material utilization and waste reduction helps maintain quality and operational reliability in production cycles.

Given these factors, the study aims to explore and develop a system that brings substantial benefits to industrial processes by improving material utilization and reducing waste. The potential cost savings, environmental benefits, and operational

efficiencies make this study highly relevant for industries looking to modernize their production processes.

1.3 OBJECTIVES OF THE STUDY

This project aims to design and implement an automated pipe-cutting optimization system that minimizes material waste while fulfilling specific cut length requirements in an industrial context. This system will analyze stock data and customer requirements to generate efficient cutting patterns, reducing the need for manual calculations and helping companies use their materials effectively. The key components of this system include:

1. **Data Acquisition Module**: This module will serve as the system's starting point, collecting data on available pipe stock and customer requirements. A user-friendly interface will allow data input through CSV or direct input, with built-in validation checks to ensure the data is complete and accurate before processing.
2. **Optimization Engine**: The core of the system, this engine will use advanced optimization algorithms, such as the Best Fit Decreasing (BFD) or other heuristic approaches, to identify optimal cut patterns. By accounting for kerf loss (3 mm in this project), the system will minimize waste while fulfilling all length requirements.
3. **Simulation & Validation Module**: Before any cutting pattern is applied in real production, this module will simulate the proposed cuts to ensure feasibility. This involves checking if the generated patterns can be practically implemented in real-world conditions, taking into account any specific constraints related to pipe handling, cutting accuracy, or operational limitations.
4. **Results Dissemination Module**: This component will present the optimized cutting patterns in an easily understandable format, utilizing visual tools such

as horizontal stacked bar charts. The bar charts will highlight the segments of each pipe used, the waste portions, and the overall utilization efficiency. Additionally, the results can be exported in multiple formats, such as CSV or PDF, for operational use and record-keeping.

5. **Feedback & Learning Module**: Continuous improvement is essential for maintaining efficiency in the long term. This module will monitor the performance of the implemented patterns and use feedback from actual production to refine the algorithm. By learning from past data, the system will improve over time, providing increasingly efficient cutting solutions that adapt to changing material stock, demands, and operational factors.

1.4 OVERVIEW OF THE PROJECT

This study has several critical objectives aimed at enhancing the efficiency of the pipe-cutting process in industrial applications:

1. **Development of an Optimization Algorithm:** The primary objective is to develop a robust algorithm capable of generating optimal cutting patterns that minimize waste. This algorithm will handle complex calculations and maximize the utility of each 6000 mm pipe by considering kerf loss and demand requirements.
2. **User-Friendly Interface and Data Management:** The study aims to create a user-friendly interface where users can input data on pipe stock and cutting requirements effortlessly. Ensuring smooth data input and management allows for accurate processing and generates reliable results without requiring extensive manual input or adjustments.
3. **Practical Cutting Patterns and Operational Feasibility:** A key objective is to ensure that the system not only provides theoretically optimal cutting patterns but also generates patterns that can be practically implemented in real-world manufacturing conditions.

4. **Data Visualization and Reporting:** The study seeks to create intuitive visualizations for the generated cutting plans. By using stacked bar graphs and other visuals, the system will clearly present the optimized cutting patterns, highlighting each cut's length and remaining waste for easy interpretation by operators.
5. **Adaptive Learning for Continuous Improvement:** Another objective is to implement a feedback loop that allows the system to learn from real-world performance and refine the optimization algorithm over time. This feature aims to make the system adaptive, improving the accuracy of its recommendations and maintaining its relevance as production needs and material availability change.
6. **Contributing to Sustainable Manufacturing Practices:** The broader objective of this study is to help industries align with sustainable practices by reducing material waste. The optimized pipe-cutting solution aligns with global sustainability efforts, making manufacturing operations more environmentally responsible and resource-efficient.

In summary, the primary goal of this study is to develop a comprehensive, automated system that not only improves efficiency and reduces waste in the pipe-cutting process but also aligns with the financial, operational, and sustainability goals of modern industrial manufacturing. The solution is expected to provide long-term benefits in terms of cost savings, productivity improvements, and alignment with global environmental standards.

CHAPTER 2

REVIEW OF LITERATURE

2.1 INTRODUCTION

The literature survey explores existing research and methodologies in the field of material optimization for manufacturing, with a focus on pipe cutting. Studies in this area highlight the significance of minimizing material waste and maximizing efficiency through various computational and heuristic approaches. Research has shown that optimization in cutting processes can significantly reduce costs and improve environmental sustainability, making it a key area of interest in industrial production.

Different algorithms, such as **Best Fit Decreasing (BFD)** and **Genetic Algorithms (GA)**, have been widely studied for their effectiveness in reducing leftover material by calculating optimal cutting patterns. Additionally, the **kerf loss** factor, representing the material lost in each cut, is an essential consideration for precise optimization, as slight adjustments in calculation can lead to substantial cost savings over time.

This survey will examine various optimization algorithms, real-world applications, and feedback mechanisms that continuously improve cutting efficiency. By reviewing existing systems and methodologies, this survey aims to identify gaps and best practices that can be applied or enhanced in the proposed solution, focusing on industrial relevance and practical adaptability.

2.2 FRAMEWORK OF PIPE CUTTING OPTIMIZATION (PCO)SYSTEM

The framework of the **Pipe Cutting Optimization System** provides a structured approach to automating and optimizing the process of cutting pipes for industrial applications. The system aims to reduce waste, maximize material utilization, and

enhance operational efficiency. The framework consists of several integrated modules, each with a specific role in achieving optimal cutting patterns.

1. Data Acquisition & Validation Module

This module serves as the starting point, allowing users to input pipe stock and cutting requirements through a user-friendly interface. The module supports both manual entry and CSV uploads, with built-in **Data Validation** to ensure that all input data is complete, accurate, and ready for further processing.

- **Data Input Interface**: Collects data on pipe lengths, quantities, and required cuts.
- **Data Validation**: Ensures accuracy in data entry, reducing errors in the optimization process.

2. Optimization Engine

The **Optimization Engine** is the core component that calculates the most efficient cutting patterns to minimize waste and fulfill cut requirements. The engine leverages algorithms like **Best Fit Decreasing (BFD)** or **Genetic Algorithms (GA)** to produce patterns that maximize the utilization of each 6000 mm pipe, while accounting for a 3 mm kerf loss (material lost per cut).

- **Algorithm Core**: Calculates optimal cutting patterns based on input data, aiming to use the fewest pipes possible.
- **Pattern Generation**: Creates cutting sequences for each pipe, ensuring minimal waste and adherence to requirements.

3. Simulation & Validation Module

This module validates the practicality of the cutting patterns generated by the Optimization Engine, simulating the cuts virtually to check for real-world feasibility.

The simulation process ensures that cutting plans align with actual manufacturing constraints, such as machine limitations and material handling needs.

- **Simulation Environment**: Tests cutting patterns virtually for feasibility.
- **Pattern Validation**: Ensures the generated patterns can be accurately implemented in a production setting.

4. Results Dissemination Module

Once validated, the cutting patterns and results are displayed in an intuitive format. This module presents data in the form of **horizontal stacked bar graphs** to visualize each cut within the pipe length, along with summaries of material usage and waste.

- **Analytics Dashboard**: Provides metrics such as waste percentage, material saved, and cost benefits.
- **Export Options**: Allows results to be exported in formats like CSV and PDF for operational use.

5. Feedback & Learning Module

The system continuously improves by analyzing the effectiveness of cutting patterns and refining its algorithms based on real-world feedback. This adaptive learning process ensures that the system evolves over time, improving cutting efficiency as production needs and material availability change.

- **Performance Monitoring**: Tracks actual results against projected efficiencies.
- **Continuous Improvement**: Uses past data to adjust the algorithm for better future performance.

CHAPTER 3

SYSTEM OVERVIEW

3.1 EXISTING SYSTEM

Current pipe-cutting processes in industrial settings often involve a combination of manual planning, heuristic approaches, and standalone optimization tools. However, these systems have several limitations:

- **Manual Planning and Errors**: Many systems rely on human operators to plan cutting patterns, which is time-consuming and prone to errors, leading to inefficient material usage and increased waste.
- **Limited Automation**: Existing systems often lack automation, requiring operators to manually input data, verify patterns, and track outcomes. This limits productivity and the ability to quickly respond to changes in production requirements.
- **Sub-optimal Material Utilization**: Basic heuristic approaches, such as simple nesting or first-fit algorithms, do not consistently achieve optimal cutting patterns, leading to higher levels of leftover material and under-utilized stock.
- **Lack of Real-Time Feedback and Adaptability**: Existing systems rarely incorporate learning or feedback mechanisms, limiting their ability to adapt to new data or improve over time.
- **Minimal Data Visualization and Reporting**: Reporting capabilities in existing systems are limited, often presenting data in basic formats that do not provide actionable insights into material utilization or cost savings.

3.2 PROPOSED SYSTEM

The **Proposed Pipe-Cutting Optimization System** is designed to overcome these limitations by introducing a fully automated, adaptive solution that enhances efficiency, reduces waste, and maximizes material usage. Key components of the proposed system include:

- **Data Acquisition & Validation Module**: A robust interface for easy data entry, supporting CSV uploads and manual input, with built-in validation to ensure data accuracy and completeness.
- **Optimization Engine**: Leveraging advanced algorithms like Best Fit Decreasing (BFD), this engine generates optimal cutting patterns that minimize waste and maximize utilization. By accounting for kerf loss and pipe dimensions, it ensures accurate cutting plans.
- **Simulation & Validation Module**: Before finalizing cutting patterns, the system simulates cuts to verify real-world feasibility. This reduces errors and ensures that cutting plans are practical and implementable in actual production environments.
- **Results Dissemination Module**: Visualizations, such as horizontal stacked bar charts, display cutting plans clearly, showing cuts, waste, and utilization. This module also provides an analytics dashboard for tracking metrics like waste percentage and cost savings, with options for exporting results in various formats.
- **Feedback & Learning Module**: Continuous improvement is achieved through a feedback mechanism that monitors real-world performance and refines the optimization algorithm based on past outcomes. This adaptive learning enables the system to improve cutting efficiency over time.

3.3 FEASIBILITY STUDY

The feasibility study examines the practicality of implementing the proposed system from technical, economic, and operational perspectives.

1. **Technical Feasibility**: The system is technically feasible due to the availability of modern optimization algorithms, data validation techniques, and visualization tools. These technologies allow for scalable development and ensure that the system can handle large datasets, adapt to varied pipe dimensions, and integrate seamlessly into existing manufacturing workflows.
2. **Economic Feasibility**: The proposed system can reduce material costs significantly by minimizing waste. While initial investment may be required for system setup and training, the long-term savings in material costs and operational efficiency are expected to outweigh these initial expenses. Additionally, by reducing waste, the system aligns with sustainability goals, which may further benefit the organization economically in the form of incentives or reduced waste management costs.
3. **Operational Feasibility**: From an operational standpoint, the system simplifies the pipe-cutting process, automating complex calculations and pattern generation, which reduces the need for manual planning and minimizes human error. Its user-friendly interface and real-time feedback allow operators to make quick adjustments, while adaptive learning ensures that the system continues to improve without frequent manual updates.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

The Pipe-Cutting Optimization System is designed for simplicity and efficiency, leveraging Python as the primary programming language for its versatility and extensive library support. Development can be done using user-friendly tools like VS Code or Jupyter Notebook. Key components include NumPy and Pandas for data processing, with a custom implementation of the Best Fit Decreasing (BFD) algorithm for optimization. Visualization is handled using Matplotlib for bar charts, with optional GUI support via Tkinter. Data storage relies on simple file-based formats like CSV or JSON, with SQLite as an optional lightweight database. The system runs locally as a standalone script, avoiding the complexity of server setups, and testing can be performed using Python's unittest module. This streamlined setup ensures easy deployment, maintainability, and effective optimization.

The system also incorporates basic simulation capabilities scripted directly in Python to validate cutting patterns and ensure practicality. Visualization outputs can be exported as PDFs using Matplotlib's savefig function, and data handling remains straightforward with built-in CSV and JSON support. For more advanced needs, SQLite offers a scalable yet lightweight database solution. The focus on local execution eliminates the need for cloud or server dependencies, making the system accessible and cost-effective for small to medium-scale applications. By emphasizing simplicity and leveraging Python’s rich ecosystem, the Pipe-Cutting Optimization System ensures efficient performance with minimal complexity, making it suitable for diverse operational requirements.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

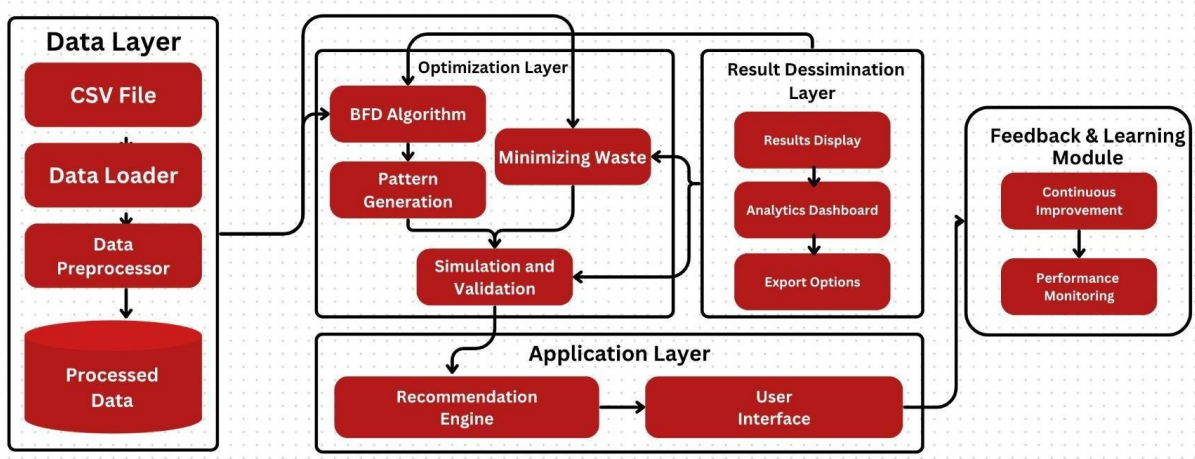


Figure 5.1: System Architecture

This system architecture outlines the key components involved in the Pipe-Cutting Optimization System, describing the function of each module and sub-module in achieving optimal cutting patterns with minimal waste.

5.2 MODULE DESCRIPTION

5.2.1. DATA ACQUISITION MODULE

This module initiates the process by capturing and validating input data. It ensures that the system has accurate and complete information on available pipe stock and required cutting specifications.

- **Data Input Interface:** This is the user-facing part of the system where users input data. Users can manually enter or upload data, such as available pipe lengths, quantities, and desired cut lengths, through an easy-to-use interface. Supported formats typically include CSV for bulk data entry.

- **Data Validation:** This sub-module verifies the accuracy and completeness of the input data. It checks for errors, such as missing values, incorrect data formats, or unrealistic values, ensuring that only clean data is passed to the optimization engine. This step prevents errors and inefficiencies in later stages of the process.

The Data Acquisition Module serves as the entry point for gathering and verifying the input data needed for the pipe-cutting optimization process. It ensures that only accurate, clean data is used in calculations, which is essential for achieving reliable and efficient cutting patterns.

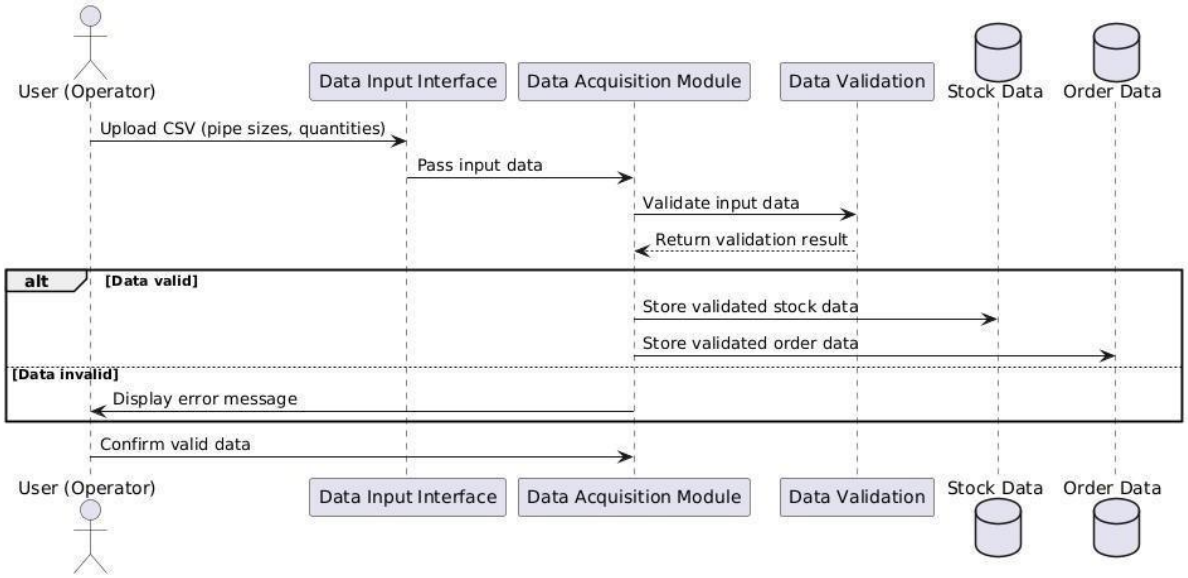


Figure 5.2: Data Acquisition Module

1. **Data Input Interface:**
    - **Function:** This interface allows users to easily upload pipe data, such as available pipe sizes and quantities, through a convenient CSV upload feature.
    - **Details:** Users can upload data files containing information about the available stock (e.g., pipe length, diameter, quantity) and cutting requirements (e.g., required lengths and quantities). The interface is designed for ease of use, minimizing user error and facilitating bulk data entry.

2. **Data Validation**:
    - **Function**: This component checks the uploaded data for completeness, consistency, and correctness before it is processed in the optimization engine.
    - **Details**: The validation step confirms that there are no missing values, incompatible formats, or unrealistic values (e.g., negative lengths or quantities). By filtering out incorrect data early, this component ensures that only high-quality data is used in generating cutting patterns, which helps prevent errors in subsequent stages and enhances the overall efficiency of the system.

5.2.2. OPTIMIZATION ENGINE MODULE

The Optimization Engine is the core of the system, where cutting patterns are calculated to maximize material usage and minimize waste. It processes validated data from the Data Acquisition Module to create optimal cutting plans.

- **Algorithm Core:** This is where the primary optimization algorithm, such as Best Fit Decreasing (BFD), is implemented. It uses mathematical computations to find the best fit for each required cut within the available stock, while considering kerf loss (material lost with each cut). The algorithm aims to minimize the number of pipes used and reduce leftover material.

- **Pattern Generation:** Based on the results of the algorithm, the Pattern Generation component creates detailed cutting sequences, specifying the placement and order of cuts for each pipe. This sub-module generates a list of cutting instructions that maximally utilizes each pipe, providing an efficient solution to the pipe-cutting problem.

The Optimization Engine is the central processing component responsible for creating optimal cutting patterns that minimize material waste and maximize efficiency. It takes validated input data and applies optimization algorithms to determine the best way to cut available pipes to meet requirements with minimal leftover material.

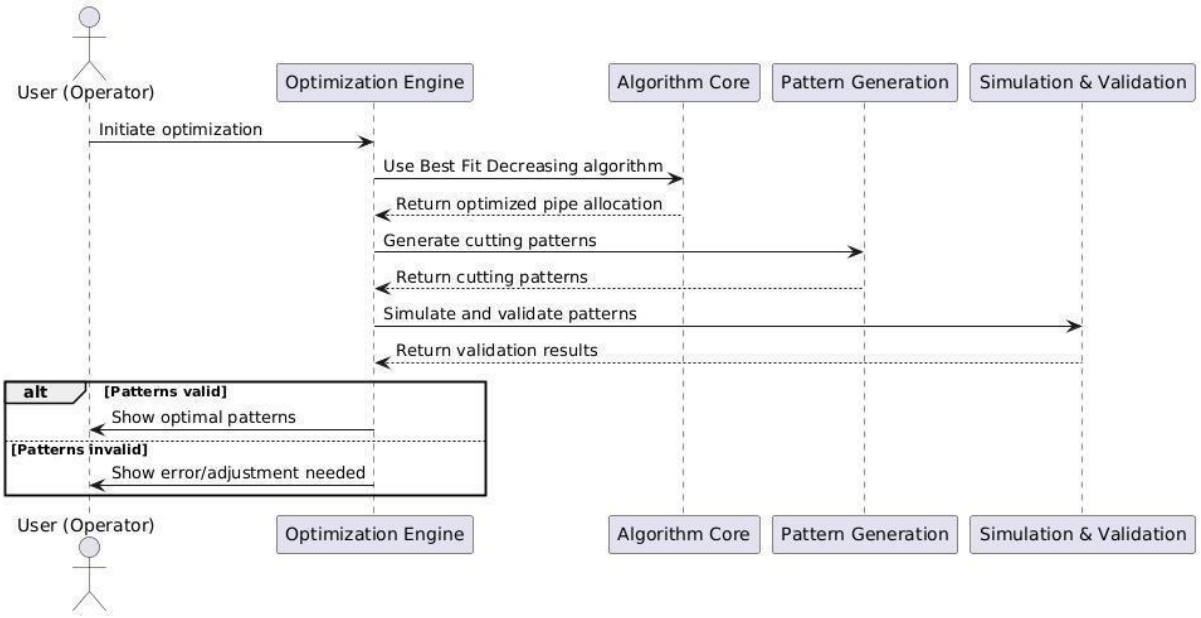


Figure 5.3: Optimization Engine Module

1. **Algorithm Core:**
    - **Function:** The Algorithm Core uses the Best Fit Decreasing (BFD) algorithm, a well-known optimization technique for minimizing waste.
    - **Details:** The BFD algorithm prioritizes larger cuts first, efficiently fitting them into available pipe lengths to minimize unused material. It allocates cuts based on available lengths, reducing the need for additional pipes. The algorithm also considers kerf (material loss due to cutting width) and any additional requirements to ensure each cut is as efficient as possible.

2. **Pattern Generation:**
    - **Function:** This component transforms the algorithm's output into detailed cutting patterns, specifying exactly how each pipe should be segmented.
    - **Details:** The Pattern Generation module organizes cuts in a sequential, easy-to-follow plan, which guides operators or automated cutters. Each cutting pattern is a practical set of instructions that maximizes the use of each pipe, showing where each required length will be cut from available stock.

5.2.3. RESULTS DISSEMINATION MODULE

The Results Dissemination Module presents the outcomes of the optimization process in a clear and actionable way, allowing users to easily interpret, analyze, and share the cutting plans. This module ensures that users can view and understand the efficiency of the generated patterns and access insights into material utilization and waste reduction.

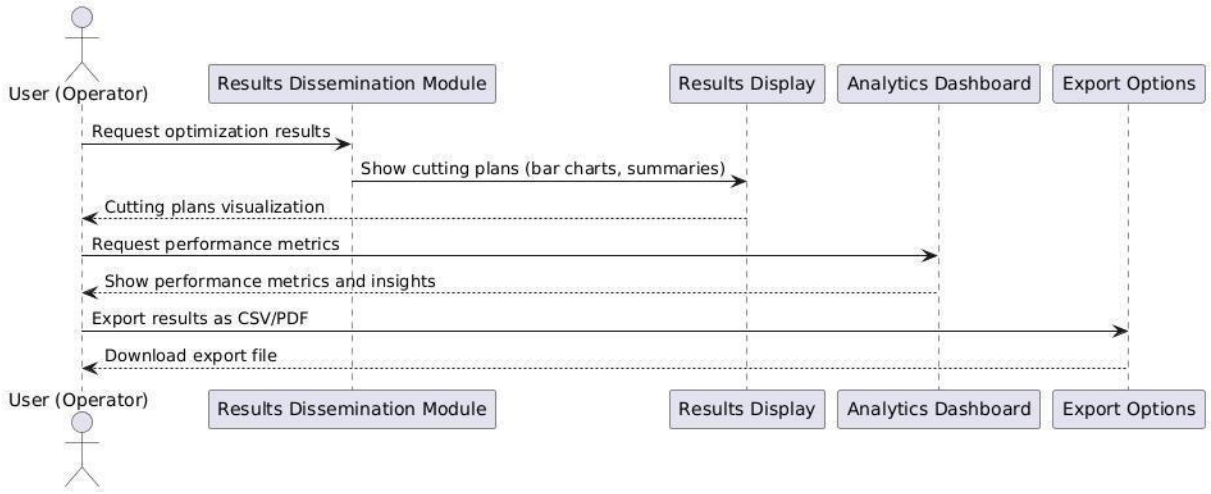


Figure 5.4: Results Dissemination Module

1. **Results Display**:
    - **Function**: This component visualizes the optimized cutting plans, making it easy for users to see how each pipe will be utilized.
    - **Details**: The display typically uses horizontal stacked bar charts that represent each pipe and show where each required cut fits within it. This graphical summary highlights material utilization, leftover portions, and any waste visually, making it straightforward for users to understand and validate the cutting patterns.

2. **Analytics Dashboard**:
    - **Function**: The Analytics Dashboard provides users with an overview of the optimization's performance metrics, including material usage efficiency and waste reduction.
    - **Details**: Key performance indicators (KPIs) like total material saved, percentage of waste reduction, and cost savings are shown here. These insights enable users to quantify the efficiency of the cutting patterns

and track improvements over time, offering valuable data for decision-making and process improvements.

3. **Export Options**:
    - **Function**: This component allows users to export cutting plans and optimization summaries in accessible formats, such as CSV or PDF, for easy sharing or integration into other systems.
    - **Details**: Users can choose to download detailed reports, including cutting plans, charts, and performance metrics, in their preferred format. This feature enhances flexibility, enabling teams to distribute results, maintain records, or perform further analysis on the exported data.

5.2.4 FEEDBACK & LEARNING MODULE

The Feedback & Learning Module continuously improves the system by learning from past performance. It adapts the optimization engine based on historical data, ensuring that cutting efficiency increases over time.

- **Performance Monitoring:** This component tracks how well each cutting plan performs in real-world operations, comparing the actual outcomes with the projected results. It captures data on metrics such as waste, material usage, and any deviations from expected performance.

- **Continuous Improvement:** Using insights from performance monitoring, the system refines its optimization algorithm. By incorporating lessons from past data, it adjusts its approach to create more efficient cutting patterns in the future, gradually improving material usage and reducing waste with each cycle.

The Feedback & Learning Module is designed to continuously enhance the optimization process by analyzing past performance and using these insights to improve future cutting plans. This module enables the system to adapt over time, making it more efficient and effective in reducing waste and maximizing material usage.

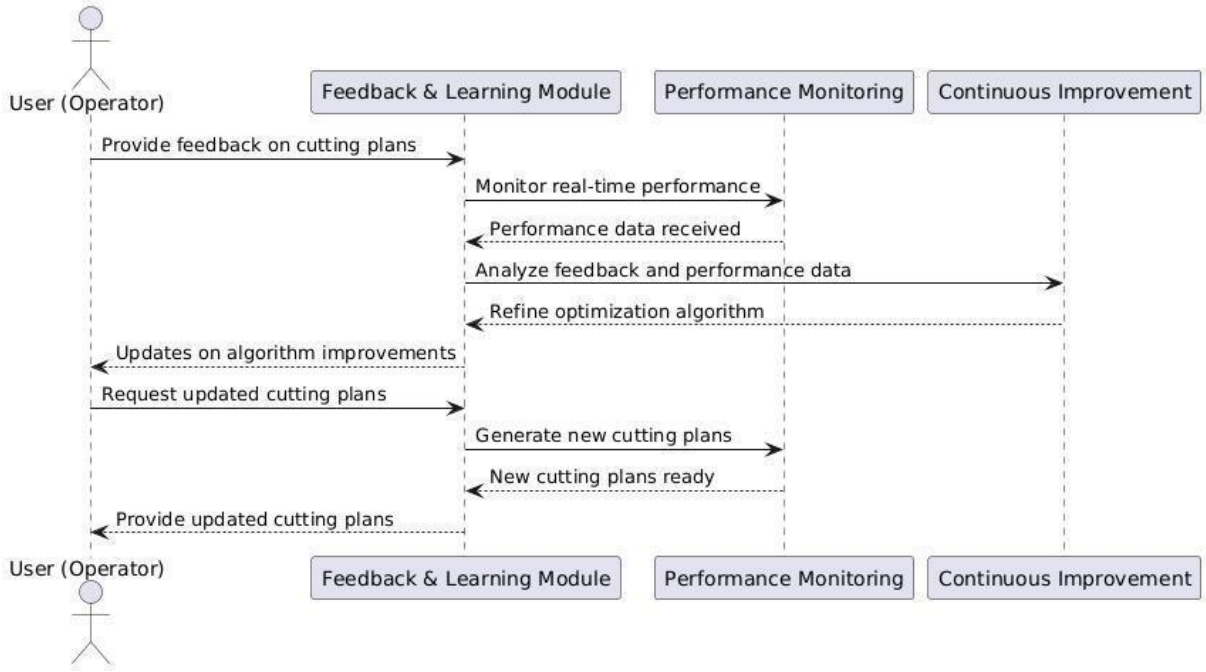


Figure 5.5: Feedback & Learning Module

1. **Performance Monitoring**:
    - **Function**: This component tracks the real-world performance of each cutting plan, comparing the optimization results with actual outcomes during implementation.
    - **Details**: Performance Monitoring collects data on metrics such as material usage efficiency, the amount of waste generated, deviations from expected cuts, and operational issues encountered during cutting. By continuously tracking this information.
2. **Continuous Improvement**:
    - **Function**: The Continuous Improvement component uses feedback from Performance Monitoring to adjust and refine the optimization algorithm, resulting in

more efficient cutting patterns over time.

- **Details**: This feedback loop leverages machine learning techniques or rule-based adjustments to improve the algorithm's accuracy and efficiency. For example, if patterns with specific configurations consistently generate more waste, the system learns to avoid these configurations in future calculations. Continuous Improvement adapts the optimization strategy to better meet the unique demands of each operation, resulting in improved material usage and minimized waste with each new iteration.

5.3 STEPS FOR BFD ALGORITHM

1. **Prepare Data**:
   - Define the raw pipe length (e.g., 10 meters).
   - List the required pipe lengths (e.g., 6m, 3m, 4m, etc.).
2. **Sort the Required Lengths**:
   - Sort the required pipe lengths from **largest to smallest**.
3. **Start Allocating Cuts**:
   - Find the raw pipe that best fits the cut (i.e., has the least leftover space).
   - If no existing pipe fits, start a new one.
4. **Update Pipe Lengths**:
   - After each cut, update the remaining length of the selected pipe.
   - Remove a pipe from the list if it’s fully used.
5. **Repeat**:
   - Continue allocating cuts until all required lengths are assigned to pipes.
6. **Result**:
   - The output will show how the cuts were made and how much scrap (leftover material) remains.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 RESULTS

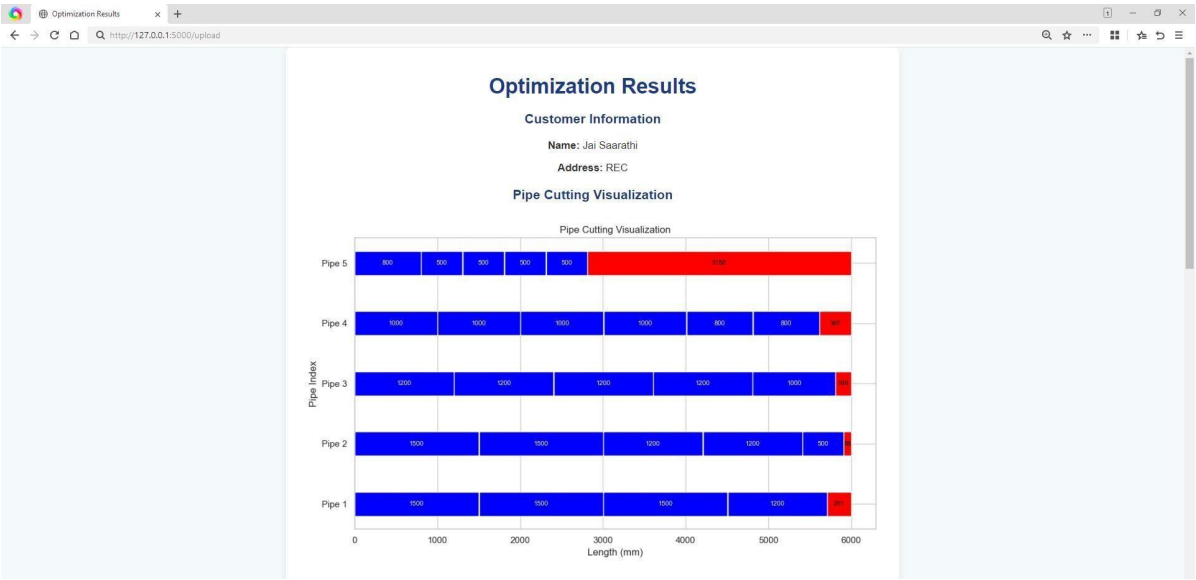


Figure 6.1: Pipe cutting visualization

The Pipe Cutting Optimization System aims to reduce material waste and maximize efficiency in industrial manufacturing. By using an advanced optimization algorithm, the system generates cutting plans that minimize leftover material, improving resource utilization. The cutting process is visualized through bar charts, showing used and unused material. The system ensures that cuts are made in the most efficient way possible, leading to cost savings and improved operational efficiency. Performance metrics such as total material used, waste percentage, and cutting loss are provided for each optimization task. The system also allows for real-time feedback and continuous improvement, ensuring long-term sustainability.

6.2 DISCUSSION

1. **Algorithm Selection:** Utilizes the Best Fit Decreasing (BFD) algorithm, which efficiently assigns pipe segments to minimize leftover lengths.

2. **Data Validation:** A Data Acquisition Module verifies completeness and correctness of input data, ensuring reliable analysis.

3. **Pattern Generation:** The Pattern Generation Engine creates optimal cutting plans for each available pipe length, aiming to minimize material waste.

4. **Simulation & Validation:** Each cutting pattern undergoes simulation to verify feasibility and practicality, reducing the risk of errors.

5. **Real-Time Performance Monitoring:** The system includes a Feedback & Learning Module to track cutting accuracy and adjust future patterns accordingly.

6. **Visual Analytics :** Displays pipe usage and efficiency metrics via bar chartsand summaries, enabling easy assessment of each cutting plan's effectiveness.

7. **Exportable Results:** Users can export cutting results in formats like CSV and PDF for convenient record-keeping and analysis.

8. **Waste Reduction Metrics:** Tracks key performance indicators related to material waste, improving insights into utilization rates.

9. **Adaptability:** Supports multiple pipe lengths and thicknesses, ensuring applicability to varied industrial scenarios.

10. **Continuous Improvement:** Feedback loops leverage performance data to refine the algorithm, enhancing cutting efficiency over time.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The pipe-cutting optimization system provides a robust solution for improving material utilization and efficiency in industrial manufacturing. By leveraging the Best Fit Decreasing algorithm and incorporating modules for data validation, pattern generation, and real-time feedback, the system ensures reliable and effective cutting plans. The visual analytics dashboard and exportable reports offer transparency and actionable insights, while continuous performance monitoring supports adaptive improvements. Overall, this system significantly reduces waste, lowers operational costs, and enables smarter resource allocation, ultimately contributing to more sustainable and efficient manufacturing processes.

7.2 FUTURE ENHANCEMENT

To enhance our pipe cutting estimation services, we plan to integrate advanced optimization algorithms to minimize waste, develop a mobile application for seamless access to reports, and introduce cloud-based analytics for real-time tracking of efficiency. Future upgrades will include integration with inventory management systems, 3D visualization of cutting layouts, and CNC machine connectivity for precise execution. Additionally, we aim to support sustainability by recommending waste reuse and calculating carbon footprint reductions, ensuring our services remain innovative, efficient, and environmentally friendly.

APPENDIX

A1.1 SOURCE CODE

1) app.py

```
from flask import Flask, request, render_template, send_from_directory
import pandas as pd
import matplotlib
import seaborn as sns
import matplotlib.pyplot as plt
import os
import numpy as np

matplotlib.use('agg') # Use the 'agg' backend for non-GUI environments

app = Flask(__name__)

available_pipe_length = 6000
cutting_loss = 3  # Loss per cut in mm

# Function to calculate the best fit decreasing algorithm
def best_fit_decreasing(available_pipe_length, sizes_quantities, cutting_loss):
    sizes_quantities.sort(key=lambda x: -x[0])  # Sort sizes in descending
order
    pipes = []
    pipe_remainings = []

    for size, quantity in sizes_quantities:
        for _ in range(quantity):
            best_fit_index = -1
            min_space_left = available_pipe_length + 1
            for i in range(len(pipes)):
                if pipes[i]['remaining'] >= size + cutting_loss and
pipes[i]['remaining'] - (size + cutting_loss) < min_space_left:
                    best_fit_index = i
                    min_space_left = pipes[i]['remaining'] - (size +
cutting_loss)
            if best_fit_index == -1:
                new_pipe = {'lengths': [size], 'remaining':
available_pipe_length - (size + cutting_loss)}
                pipes.append(new_pipe)
                pipe_remainings.append(available_pipe_length - (size +
cutting_loss))
            else:
                pipes[best_fit_index]['lengths'].append(size)
                pipes[best_fit_index]['remaining'] -= (size + cutting_loss)
```

```
                pipe_remainings[best_fit_index] -= (size + cutting_loss)

    return pipes, pipe_remainings, len(pipes)


# Visualization functions
def visualize_pipes(pipes, available_pipe_length, cutting_loss):
    sns.set(style="whitegrid")
    fig, ax = plt.subplots(figsize=(10, 6))
    bar_height = 0.4

    total_used_length = 0
    total_waste = 0
    for i, pipe in enumerate(pipes):
        cuts = pipe['lengths']
        used_length = sum(cuts) + cutting_loss * (len(cuts) - 1)
        unused_length = available_pipe_length - used_length
        total_used_length += used_length
        total_waste += unused_length

        bottom = 0
        for cut in cuts:
            ax.barh(i, cut, bar_height, left=bottom, color='blue')
            ax.text(bottom + cut / 2, i, f'{cut}', ha='center', va='center',
color='white', fontsize=8)
            bottom += cut + cutting_loss

        ax.barh(i, unused_length, bar_height, left=used_length, color='red')
        ax.text(used_length + unused_length / 2, i, f'{unused_length}',
ha='center', va='center', color='black', fontsize=8)

    ax.set_ylabel('Pipe Index')
    ax.set_xlabel('Length (mm)')
    ax.set_title('Pipe Cutting Visualization')
    ax.set_yticks(range(len(pipes)))
    ax.set_yticklabels([f'Pipe {i+1}' for i in range(len(pipes))])

    plt.tight_layout()
    img_path = 'static/pipe_cutting_visualization.png'
    plt.savefig(img_path)
    plt.close()

    return img_path, total_used_length, total_waste


def cutting_efficiency(total_used_length, total_waste, total_length):
    efficiency = (total_used_length / total_length) * 100
    return efficiency
```

```
@app.route('/')
def index():
    return render_template('index.html')


@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part'

    file = request.files['file']
    if file.filename == '':
        return 'No selected file'

    if file:
        customer_name = request.form.get('name')
        customer_address = request.form.get('address')

        os.makedirs('uploads', exist_ok=True)
        file_path = os.path.join('uploads', file.filename)
        file.save(file_path)

        data = pd.read_csv(file_path)
        sizes_quantities = list(zip(data['Size'], data['Quantity']))
        pipes, pipe_remainings, total_pipes_used =
best_fit_decreasing(available_pipe_length, sizes_quantities, cutting_loss)

        img_path, total_used_length, total_waste = visualize_pipes(pipes,
available_pipe_length, cutting_loss)

        total_length = available_pipe_length * total_pipes_used
        efficiency = cutting_efficiency(total_used_length, total_waste,
total_length)

        # Create a histogram for cut distribution using matplotlib
        cuts = [cut for pipe in pipes for cut in pipe['lengths']]
        plt.hist(cuts, bins=np.arange(min(cuts), max(cuts) + 50, 50),
color='skyblue', edgecolor='black')
        plt.xlabel("Cut Size (mm)", fontsize=14)
        plt.ylabel("Frequency of Cuts", fontsize=14)
        plt.title("Cut Distribution Histogram", fontsize=16)
        cut_dist_img = 'static/cut_distribution.png'
        plt.savefig(cut_dist_img)
        plt.close()

        # Create a pie chart for waste vs. used material
        labels = ['Used Material', 'Waste']
        sizes = [total_used_length, total_waste]
```

```
        plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
        waste_pie_img = 'static/waste_pie_chart.png'
        plt.savefig(waste_pie_img)
        plt.close()

        # Create a histogram for cut size distribution using matplotlib
        plt.hist(cuts, bins=np.arange(min(cuts), max(cuts) + 50, 50),
color='salmon', edgecolor='black')
        plt.xlabel("Cut Size (mm)", fontsize=14)
        plt.ylabel("Frequency", fontsize=14)
        plt.title("Cut Size Distribution", fontsize=16)
        cut_size_dist_img = 'static/cut_size_distribution.png'
        plt.savefig(cut_size_dist_img)
        plt.close()

        return
            render_template('re
            sult.html',
            img_path='pipe_cutting_visualization.png',
            name=customer_name,
            address=customer_address,
            total_pipes=total_pipes_used,
            efficiency=efficiency,
            cut_dist_img='cut_distribution.png',
            waste_pie_img='waste_pie_chart.png',
            cut_size_dist_img='cut_size_distribution.png'
        )


@app.route('/static/<path:filename>')
def serve_static(filename):
    return send_from_directory('static', filename)
if__name__== '__main__':
    app.run(debug=True)
```

2) optimization.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Constants
AVAILABLE_PIPE_LENGTH = 6000  # in mm
CUTTING_LOSS = 3  # kerf loss in mm

def best_fit_decreasing(available_pipe_length, sizes_quantities, cutting_loss):
```

```
    sizes_quantities.sort(key=lambda x: -x[0]) # Sort sizes in descending
order
    pipes = []
    pipe_remainings = []

    for size, quantity in sizes_quantities:
        for _ in range(quantity):
            best_fit_index = -1
            min_space_left = available_pipe_length + 1
            for i in range(len(pipes)):
                remaining = pipes[i]['remaining']
                if remaining >= size + cutting_loss and remaining - (size +
cutting_loss) < min_space_left:
                    best_fit_index = i
                    min_space_left = remaining - (size + cutting_loss)
            if best_fit_index == -1:
                new_pipe = {'lengths': [size], 'remaining':
available_pipe_length - (size + cutting_loss)}
                pipes.append(new_pipe)
                pipe_remainings.append(new_pipe['remaining'])
            else:
                pipes[best_fit_index]['lengths'].append(size)
                pipes[best_fit_index]['remaining'] -= (size + cutting_loss)
                pipe_remainings[best_fit_index] -= (size + cutting_loss)

    return pipes, pipe_remainings, len(pipes)

def visualize_pipes(pipes, available_pipe_length, cutting_loss):
    # Set the figure size (increase height)
    fig, ax = plt.subplots(figsize=(12, 12))
    bar_height = 0.8  # Bar height, adjust if needed

    for i, pipe in enumerate(pipes):
        cuts = pipe['lengths']
        used_length = sum(cuts) + cutting_loss * (len(cuts) - 1)
        unused_length = available_pipe_length - used_length

        bottom = 0
        for cut in cuts:
            ax.barh(i, cut, bar_height, left=bottom, color='blue')
            bottom += cut + cutting_loss

        # Plot the unused space
        ax.barh(i, unused_length, bar_height, left=used_length, color='red')

    ax.set_xlabel('Length (mm)')
    ax.set_ylabel('Pipe Index')
    ax.set_title('Pipe Cutting Visualization')
```

```
    plt.tight_layout()
    img_path = 'static/pipe_cutting_visualization.png'
    plt.savefig(img_path)
    plt.close()
    return img_path

def load_and_prepare_data(file_path):
    data = pd.read_csv(file_path)
    sizes_quantities = list(zip(data['Size'], data['Quantity']))
    return sizes_quantities
```

3) index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Pipe Cutting Optimizer</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Pipe Cutting Optimizer</h1>

        <form id="upload-form" action="{{ url_for('upload_file') }}"
method="post" enctype="multipart/form-data">
            <label for="name">Customer Name:</label>
            <input type="text" id="name" name="name" required><br><br>

            <label for="address">Customer Address:</label>
            <input type="text" id="address" name="address" required><br><br>

            <!-- Drag-and-drop area -->
            <div id="drop-zone" class="drop-zone">
                Drag & Drop your CSV file here, or click to select
            </div><br><br>

            <!-- Hidden file input field that gets triggered by the drop zone
-->
            <input type="file" name="file" id="file-input" accept=".csv"
required style="display:none;"><br><br>

            <button type="submit">Upload and Optimize</button>
```

```
        </form>

        <script>
            // JavaScript for drag-and-drop functionality
            const dropZone = document.getElementById('drop-zone');
            const fileInput = document.getElementById('file-input');

            // Handle click to open file selector
            dropZone.addEventListener('click', () => fileInput.click());

            // Change appearance on drag-over
            dropZone.addEventListener('dragover', (e) => {
                e.preventDefault();
                dropZone.classList.add('dragover');
            });

            // Revert appearance when drag leaves
            dropZone.addEventListener('dragleave', () =>
dropZone.classList.remove('dragover'));

            // Handle dropped files
            dropZone.addEventListener('drop', (e) => {
                e.preventDefault();
                dropZone.classList.remove('dragover');
                const files = e.dataTransfer.files;

                // Validate file type
                if (files.length > 0 && files[0].type === 'text/csv')
                    { fileInput.files = files; // Assign the dropped file to the
file input
                    dropZone.textContent = files[0].name;                 }
else {
                    alert("Please drop a CSV file.");
                }
            });

            // Update drop-zone text when file is selected through file input
            fileInput.addEventListener('change', () => {
                if (fileInput.files.length > 0)
                    { dropZone.textContent =
                    fileInput.files[0].name;
                }
            });
        </script>
    </div>
</body>
</html>
```

4) result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Optimization Results</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Optimization Results</h1>

        <h3>Customer Information</h3>
        <p><strong>Name:</strong> {{ name }}</p>
        <p><strong>Address:</strong> {{ address }}</p>

        <h3>Pipe Cutting Visualization</h3>
        <img src="{{ url_for('serve_static', filename=img_path) }}" alt="Pipe
Cutting Visualization" class="result-img">

        <h3>Cut Distribution Histogram</h3>
        <img src="{{ url_for('serve_static', filename=cut_dist_img) }}"
alt="Cut Distribution Histogram" class="result-img">

        <h3>Waste vs. Used Material (Pie Chart)</h3>
        <img src="{{ url_for('serve_static', filename=waste_pie_img) }}"
alt="Waste vs Used Material Pie Chart" class="result-img">

        <h3>Cutting Efficiency: {{ efficiency }}%</h3>

        <h3>Total Pipes Used: {{ total_pipes }}</h3>

        <br><br>
        <button href="{{ url_for('index') }}" type="submit">Upload Another
File</button>
    </div>
</body>
</html>
```

5) styles.html

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f9fc; /* Light blue-gray background */
    color: #333;
    text-align: center;
}
.container {
    max-width: 900px;
    margin: 0 auto;
    padding: 20px;
    background-color: white; /* White background for the content */
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

h1 {
    color: #1e3a8a; /* Dark blue color */
    font-size: 32px;
    margin-bottom: 20px;
}

h3 {
    color: #1e3a8a;
    margin-top: 20px;
    font-size: 20px;
}

/* Styling for the drag-and-drop area */
.drop-zone {
    width: 95%;
    height: 200px;
    padding: 20px;
    margin: 10px 0;
    background-color: #e2e8f0; /* Light gray background */
    border: 2px dashed #1e3a8a; /* Blue dashed border */
    border-radius: 8px;
    color: #1e3a8a; /* Blue text */
    font-size: 25px;
    cursor: pointer;
    display: flex;
    justify-content: center;
    align-items: center;
    text-align: center;
}
```

```css
/* Change the drop zone appearance on drag-over */
.drop-zone.dragover {
    background-color: #cbd5e1; /* Slightly darker gray background */
    border-color: #2563eb; /* Darker blue border */
}

/* Input field and button styling */
input[type="text"], input[type="file"], button {
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ddd;
    border-radius: 4px;
    width: 95%;
    font-size: 16px;
}

button {
    background-color: #1e3a8a;
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #2563eb;
}

/* Button styling for hover */
button[type="submit"]:hover {
    background-color: #2563eb; /* Blue background on hover */
}

@media screen and (max-width: 768px) {
    .container {
        padding: 10px;
    }

    h1 {
        font-size: 24px;
    }

    h3 {
        font-size: 18px;
    }
}
img{
    width: 900px;
}
```

A1.2 SCREEENSHOTS

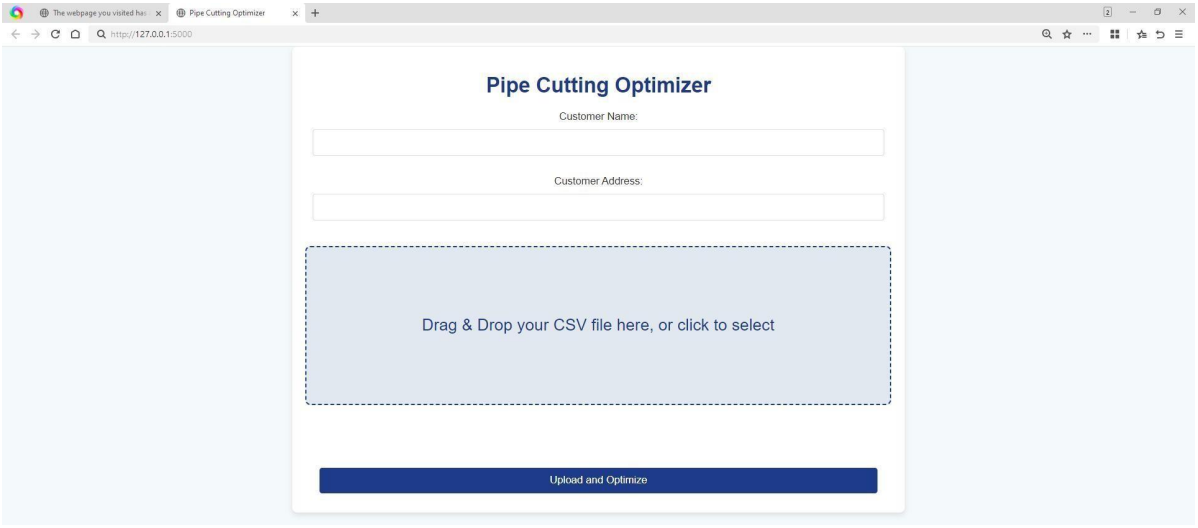


Fig A1.1: Output Interface

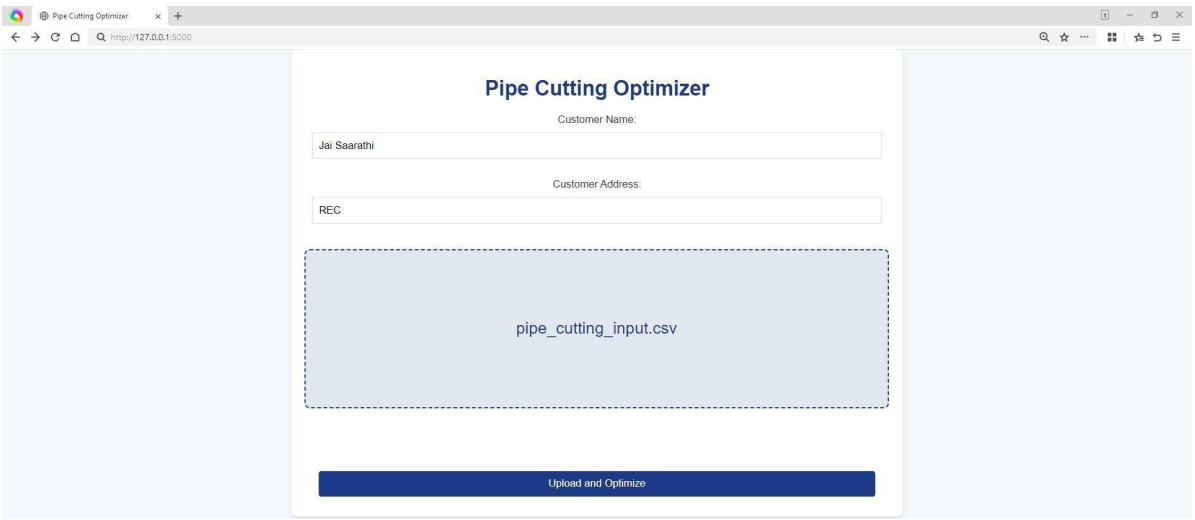


Figure A1.2: Output interface after the input is given

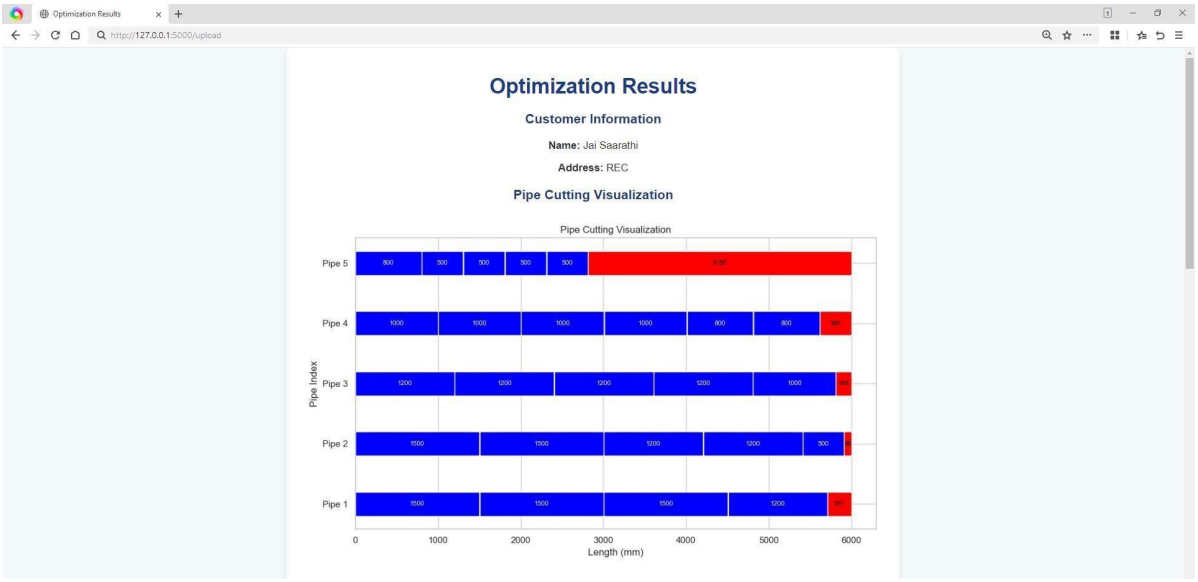


Figure A1.3: Pipe cutting visualization

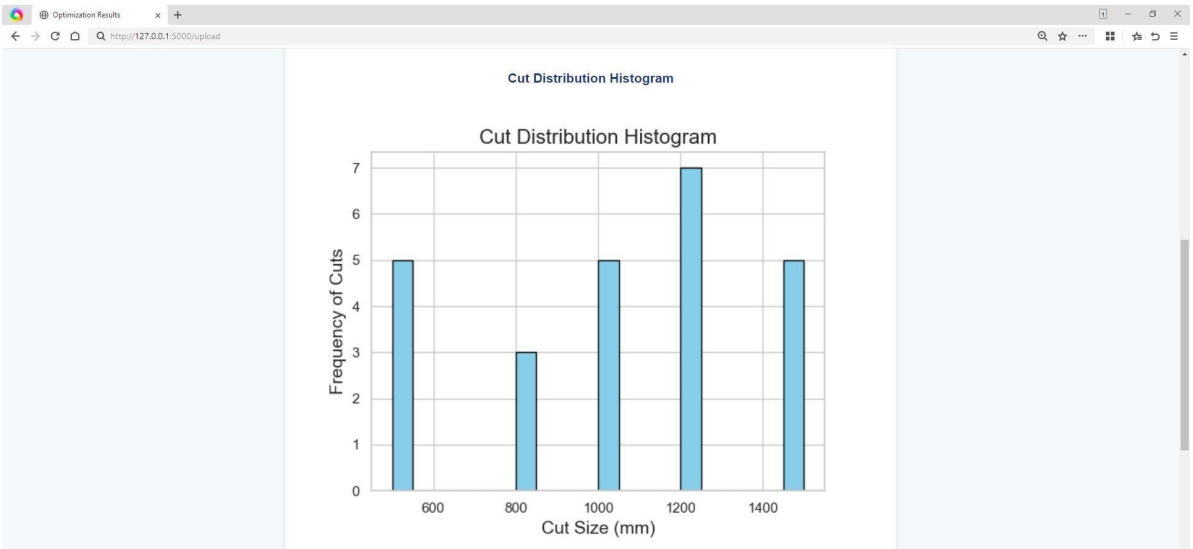


Figure A1.4: Cutting Distribution Histogram

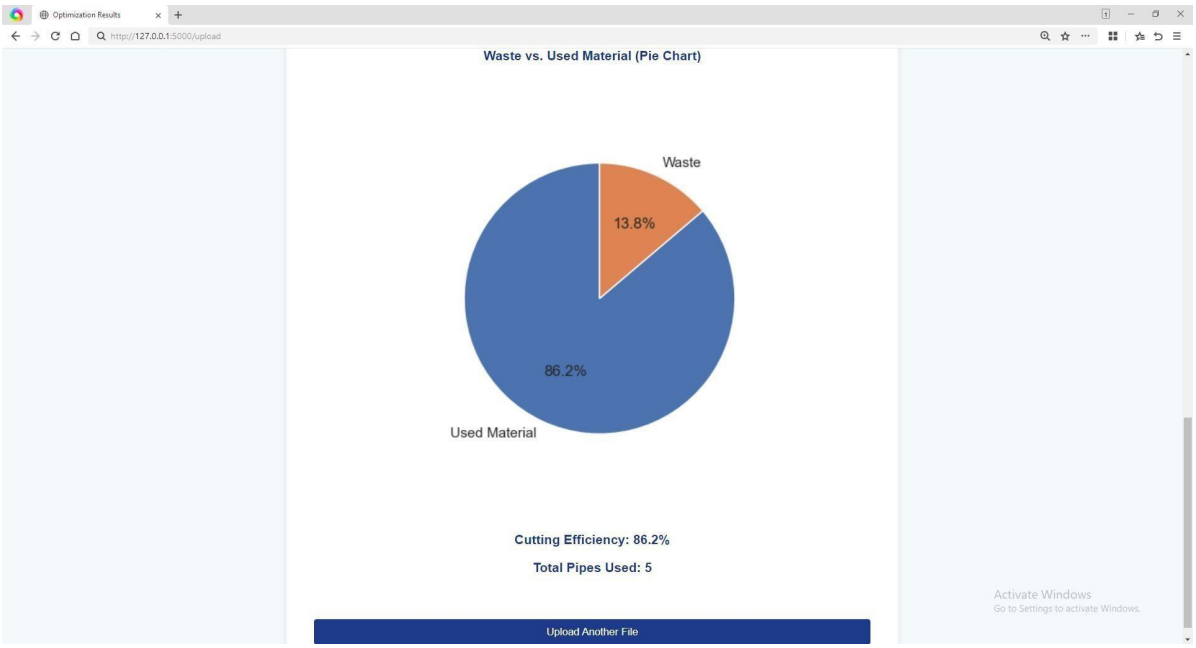


Figure A1.5: Pipe distribution

REFERENCES

[1] L. Martinez and T. Barman, "Optimization Techniques in Manufacturing: An Overview," *Journal of Industrial Engineering and Management*, vol.12, no. 4,pp. 256-267, 2021.

[2] J. Smith and Z. Hu, "Industrial Applications of Best Fit Algorithms in Manufacturing," *International Journal of Production Research*, vol. 58, no.6, pp. 1032-1045, 2020.

[3] A. Singh and S. Grover, "Material Utilization in Pipe Cutting Using Computational Approaches," *Manufacturing Technology Today*, vol. 32,no.3, pp. 188-195, 2019.

[4] R. Brown, "Inventory Management in Modern Manufacturing Systems," *Journal of Operations Management*, vol. 15, no. 2, pp. 134-145, 2018.

[5] S. Lee, "Adaptive Algorithm Approaches for Manufacturing Optimization," *Journal of Intelligent Manufacturing*, vol. 28, no. 5, pp. 881-892, 2020.

[6] Y. Wang and X. Zhao, "Real-Time Data Analytics for IoT-Enabled Manufacturing Systems," *Sensors*, vol. 22, no. 8, p. 3334, 2022.

[7] E. Johnson and C. Wu, "Machine Learning Applications in Industrial Optimization," *Journal of Manufacturing Systems*, vol. 36, no. 5, pp. 1011-1023, 2023.

[8] P. Thomas and M. Richardson, "Integrating ERP Systems in Manufacturing for Enhanced Workflow," *International Journal of Production Economics*, vol. 41, no. 2, pp. 210-219, 2021.

[9] N. Patel, "Predictive Analytics in Manufacturing: Enhancing Operational Efficiency," *Operations Research Perspectives*, vol. 23, no. 4, pp. 273-286, 2019.

[10] L. Chen and D. Smith, "Sustainability Tracking in Industrial Processes: Methods and Applications," *Journal of Cleaner Production*, vol. 250, p. 119735, 2020.