ROUTE OPTIMIZATION SYSTEM FOR WAREHOUSE ORDER PICKING USING REAL-TIME PATH PLANNING AND SHELF SELECTION

Palihena P.D.M.P
(IT21079672)

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

ROUTE OPTIMIZATION SYSTEM FOR WAREHOUSE ORDER PICKING USING REAL-TIME PATH PLANNING AND SHELF SELECTION

Palihena P.D.M.P
(IT21079672)

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

DECLARATION

I declare that this is my work. This proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning. To the best of my knowledge and belief, it does not contain any previously published material written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
Palihena P.D.M.P	IT21079672	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: ________________________ Date _____________

Signature of the co-supervisor: ____________________ Date _____________

ABSTRACT

The order-picking process is painstakingly labor-intensive and costly in the municipal operations of warehouses. It is far too common that paths for workers or forklifts to pick multiple items across an extensive warehouse space led to much longer travel times, increased operational costs, and fatigue for workers. This project aims to develop a warehouse-based order-picking route optimization system inspired by the principles of Industry 4.0, which would allow for tracking, dynamic path-finding algorithms, and collision avoidance instructions to be incorporated into warehouse workflow operations.

The proposed system will optimize order-picking-type tasks, and evaluate and provide the shortest path through a warehouse layout based on a combination of Travelling Salesman Problem (TSP) sequencing and A-pathfinding*, that will utilize internally-developed real-time cloud-based mapping updates that allow for the routing to be adjusted for changes in delay (i.e. congestion, flagged maintenance zones, or blocked aisles). A visualization module will illustrate the suggested optimal routes for picking tasks, as well as provide the logistics managers timely decision support and monitoring.

The value of the system is its dynamic optimization, by being developed to allow for real-time collision avoidance and being able to be 'flagged for maintenance', improves on traditional static routing systems as its adaptive (not predetermined). By optimizing to eliminate unnecessary distance and wait, the system would facilitate added efficiency, improve safety, and positively increase operation throughput in any warehouse environment.

Keywords: Warehouse Management System, Order Picking Optimization, Route Planning, Traveling Salesman Problem (TSP), A* Pathfinding Algorithm, Dynamic Path Adjustment, Collision Avoidance, Industry 4.0, Real-Time Visualization, Logistics Efficiency

ACKNOWLEDGEMENT

First I would like to express my deepest gratitude to my supervisor, **Prof. Samantha Rajapaksha**, for his continuous guidance, encouragement, and invaluable feedback throughout this project. I also wish to thank **Dr. Dinuka Wijendra** for his support and constructive suggestions that helped refine this work. Finally, my sincere appreciation goes to the **academic staff of SLIIT – Faculty of Computing, Department of Information Technology**, for providing the knowledge, resources, and infrastructure necessary to complete this research.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
JWT	JSON Web Token
API	Application Programming Interface
TSP	Travelling Salesman Problem
UI	User Interface
Flask	A lightweight Python web framework used to develop web applications and APIs.
UX	User Experience
IoT	Internet of Things – a network of physical devices embedded with sensors, software, and connectivity to collect and exchange data.
A*	A-Star Pathfinding Algorithm
WMS	Warehouse Management System

1 List of abbreviations

1. INTRODUCTION

Warehouses are an essential part of the global supply chain process by providing storage, handling, and distribution capabilities for goods, raw materials, and all-important stock. In warehouse operations, order picking is often the most labor intensive and cost-influencing activity representing more than 50% total operational costs in a growing number of facilities. The use of inefficient picking routes or poorly organized item locations in warehouses can lead to unnecessary travel distance, longer time to complete an order, higher operational costs, and more fatigue for the worker or forklift operator completing the order. As goods-in-process warehouses grow in size and complexity these inefficiencies become more pronounced which can decrease overall throughput and competitiveness.

Historically, warehouses use static picking strategies, including manual assigned picking, zone picking, and simple heuristics (including nearest-neighbor routing). While static or basic picking strategies form a good basis for organizational structure and planning, they do not provide the flexibility necessary to mitigate real-world conditions such as congested aisles, blocked paths, maintenance practices, and dynamically changing stock levels. Hence, it is no surprise that many existing warehouse management systems possess limited capability to simultaneously optimize the sequence of items being picked and the path the picker takes through the warehouse layout.

Industry 4.0 technologies have brought new solutions to the existing inefficiencies resulting from ill-designed warehouse layouts. The various functions including intelligent algorithms, real time tracking and data producing decision support can now be combined. This project presents a route optimization system designed specifically for the downstream optimization of warehouse order picking. A hybrid approach to order picking will be used by implementing the use of a Travelling Salesman Problem (TSP) function to create an optimal sequence of pick locations and A pathfinding* function to create the shortest paths possible through the warehouse grid based on obstacles such as, shelves, forklifts, and flagged maintenance zones and appropriate travel paths for delivery personnel. Complex diagrams demonstrating the optimized route will be included in a visualization function for warehouse managers and personnel.

The proposal for the route and path optimization system provides advantages over the traditional static routes in place today with the use of dynamic optimization, collision avoidance and path modification around obstacles. The system expects to enhance efficiencies and reduce delays, while safely reducing unnecessary travel and increasing throughput of modern warehouses. This project is a comprehensive pathway, formulated from this research, to smart warehousing applications and most feasibly aligns and links with some of the emerging trends associated with Industry 4.0, and capable of being incorporated to existing warehouse management systems, at low cost and great return.

1.1 Background

Order picking is still one of the most labor-intensive and cost-driving activities in warehouse management; it typically represents by far the largest portion of operational costs in many facilities, over 50% in some cases [1], [12]. As warehouses become larger and layouts become more complicated, inefficient order-picking routes lead to more time spent traveling, increased costs, employee fatigue, and less throughput overall [2], [3]. Traditional approaches to order-picking such as random storage, dedicated storage, or simple heuristics usually do not lead to the least traveling distance strategy for picking items or respond to dynamic conditions such as congestion, blocked aisles, maintenance activity, etc. [10], [11].

Rapid advancements in Industry 4.0 technologies enable us to move towards automation, smart routing, and real-time decision support in warehouses [4], [5]. These advancements are underpinned by research on automated storage and retrieval systems (AS/RS) [6] or Internet of Things enabled Warehouse Management Systems (WMS) [13] that create digital insight into inventories and tracking movements. Nonetheless, there is still one fundamental issue remaining; how to dynamically optimize pick routes in real-world scenarios.

Computer science and operations research have pioneered several pathfinding and route optimization algorithms that inform today's logistics systems. Dijkstra's algorithm [7], A* [8], and Ant Colony Optimization (ACO) [9] are a few foundational algorithms that have been adapted and implemented within logistics and warehouse domains in a variety of ways with inherent trade-offs for computational cost, scalability, and for dynamic environments. More recent developments like reinforcement learning [16] and digital twin models [17] demonstrate the continued evolution towards more intelligent and adaptive approaches for route optimization.

In light of contemporary practices used hencefar, this project presents a warehouse optimized route system using TSP-based sequencing when combined with A*-based pathfinding, enhanced with obstacle awareness and visualization. The system incorporates both sequence optimization (optimizing the shortest order picking path across a number of distinct locations) as well as dynamic path planning (the shortest route through the grid layout), which combined address identified gaps from traditional order-picking by implementing systemically to capitalize on Industry 4.0 principles. The output will provide a cost-efficient approach to adaptive and scalable systems for improved efficiency, safety, and throughput in modern warehouse environments.

1.2 Literature Survey

This section reviews prior research across four major components relevant to the proposed system: order picking optimization, Industry 4.0 in warehouse management, pathfinding algorithms, and dynamic routing & collision avoidance.

1.2.1 Order Picking Optimization

Across many studies, provided evidence demonstrates that order picking consumes the most resources in a warehouse [1], [12]. Reviews completed by De Koster et al. [1] and Gu et al. [2] drew attention to route optimization as an aid to truncate the distance traveled in an order picking task. Case studies, e.g., Ankor's warehouse [3], provided insights into the possible benefits resulting in improvement of the response time in real-world scenarios. In addition, hybrid approaches (heuristics + mathematical optimization) have been utilized which yield better trade-offs between efficiency and feasibility [10], [11]. Evidence provides support for moving from static systems meaning the common monotone procedures within warehouses to dynamic and adaptive systems.

1.2.2 Industry 4.0 and Smart Warehouse Systems

The emergence of Industry 4.0 is changing global warehouse operations, requiring adaptation to technology developments including IoT, WMS, and real-time data analytics [4], [5]. Detailed studies on IoT-based warehouse tracking [13], as well as studies on warehouse management with digital twins [17], [18], indicate how to exploit data for real-time decision making. Big data analytics also provides the ability to analyze firms' dynamic capabilities [17], as they relate directly to optimizing operations and performance. These studies identify an increasing trend towards digitally based warehouse management as data-driven, with route optimization an important aspect.

1.2.3 Pathfinding and Routing Algorithms

Classic path-finding algorithms, such as Dijkstra's algorithm [7] and A*, are the foundation of warehouse routing; they offer efficient and effective shortest-path solutions on either graphs or grids. The well-known pathfinding solutions above can be expanded to a large-scale dynamic environment with more advanced solutions, including Ant Colony Optimization (ACO) [9] or heuristic planning algorithms [14]. Moreover, recent advances in deep reinforcement learning [16] are promising for dynamic order-picking environments by allowing systems to adapt their routes in real-time. These studies provide us the methodological foundation for combining TSP sequencing with pathfinding with A*.

1.2.4 Dynamic Routing and Collision Avoidance

Collision avoidance and dynamic adaptation is an important consideration from a practical standpoint in real warehouse operations with changing objects in the environment like workers, forklifts, and obstacles. The Dynamic Window Approach (DWA) for collision avoidance in robots is addressed by [15]. Looking at human–robot interaction studies [16] on safety and usability of a shared workspace also informs similar concern. Digital twin frameworks [17], [18] provide additional opportunities for simulation-based testing of collision scenarios. These considerations have laid the groundwork for including real-time obstacle-aware routing capabilities in warehouse picking optimization systems.

1.3 Research Gap

While warehouse route optimization has been widely covered in the areas of logistics and operations research, the practical application of route optimization methods in the word of warehousing is sparse and underdeveloped. If we examine existing literature, many studies solely focus on order-picking efficiency either through static heuristics or abstract optimization models. Most of the existing studies do not consider a range of factors such as dynamic pathfinding, obstacle avoidance, and real-time adaptability. In reviewing the literature we identified several key gaps that we wish to fill with this project.

1.3.1 Order picking optimization lacks real-time adaptability

A large majority of research in order picking area has been narrowly focused on either reducing travel distance or finding batching strategies with static layouts [1], [2], [12]. These strategies are arguably efficient for optimization purposes; however, they do not focus on real-time problems like congestion, aisle blockages, or maintenance zones. This is counterproductive because the practical context in which the optimization undergoes real time planning and optimizing practices create a disconnect for optimization from the modes of traffic and layouts of warehouses that are continually changing the flow and patterns of shopping and order picking activity [10], [11]..

1.3.2 Industry 4.0 adoption is limited in routing systems

As Industry 4.0 technologies such as IoT-enabled WMS and digital twins [4], [5], [13], [17], [17], [18] have started to be integrated into warehouse management, it should be noted that these technologies have only been used minimally in route optimization systems. In other words, a number of the modern-day applications are using tracking and monitoring to provide some tracking and monitoring functions, but there are very few examples of the use of IoT and digital twin data streams to dynamically recalculate the optimal picking routes. Not only does this indicate a missed opportunity to pair real-time visibility to the use of intelligent path-finding capabilities, it indicates smarter warehouse operations can benefit from real-time data and insights.

1.3.3 Classic algorithms are underutilized in warehouse-specific contexts

Fundamental algorithms, such as Dijkstra's shortest path [7], A* [8], and Ant Colony Optimization (ACO) [9], have been widely validated in graph theory and robotics. The use of these algorithms in warehouse order-picking settings; however, is limited, particularly in hybrid algorithms where sequencing (e.g., Traveling Salesman Problem) and pathfinding (A*) are intertwined. Research tends to treat the two problems as separate, rather than create a framework where one can sequence the shelves and compute paths while taking obstacles into account.

1.3.4 Collision avoidance is not sufficiently addressed

Collisions in robotics are a central area of research, taking many forms and recognizably using wildly different methods such as the Dynamic Window Approach [15] and human–robot interaction methods [16]. In considering warehouse routing, though, research methods to enhance order-picking and routing seldom considers such safety features. This is notable because a safety dimension of work in warehouses is always overlapping through the route of workers, and thus static routing is unsafe in practice, especially in dynamic environments.

1.3.5 Lack of integrated, dynamic, and visualized routing frameworks

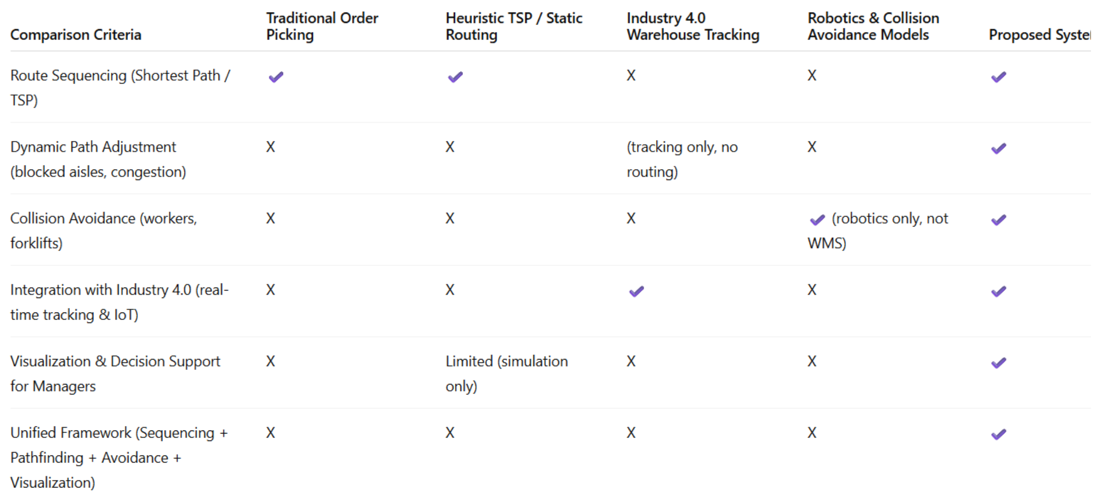
While there are many examples of order picking [1], [3], optimization heuristics [10], and tracking technologies [13], there is no integrated system that combines:

- Sequencing optimization (TSP),
- Path finding (A*)
- Dynamic adaptation (congestion, blocked aisles)

• Collision avoidance, and

• Visualization for decision support.

Most existing systems are theoretical, static, or piecemeal and not applicable directly in industrial warehouse environments. This project looks to fill the gap by providing a real-time route optimization system that is visualization-supported and develop a real-world prototype based on academic findings and algorithmic research, that can be used operationally within an Industry 4.0 warehouse delivery environment.



Comparison Criteria	Traditional Order Picking	Heuristic TSP / Static Routing	Industry 4.0 Warehouse Tracking	Robotics & Collision Avoidance Models	Proposed Syste
Route Sequencing (Shortest Path / TSP)	✓	✓	X	X	✓
Dynamic Path Adjustment (blocked aisles, congestion)	X	X	(tracking only, no routing)	X	✓
Collision Avoidance (workers, forklifts)	X	X	X	✓ (robotics only, not WMS)	✓
Integration with Industry 4.0 (real-time tracking & IoT)	X	X	✓	X	✓
Visualization & Decision Support for Managers	X	Limited (simulation only)	X	X	✓
Unified Framework (Sequencing + Pathfinding + Avoidance + Visualization)	X	X	X	X	✓

Figure 1 Research Gap

1.4 Research Problem

Order picking is one of the most resource-intensive, cost-driving activities in warehouse management. In many logistics environments, it comprises nearly 50% to 60% of total operational costs [1], [2]. Despite the importance of order picking, most existing systems still rely on static routing methods or the manual knowledge of the operator, often leading to long travel distances, poor routing, and unnecessary operator fatigue. Conventional methods, such as basic nearest-neighbor routing or static picking order sequences, are insufficiently adaptive to real-time warehouse dynamics where, for instance, aisles may be blocked, warehouses may experience traffic congestion, or a considerable delay may happen along pre-planned routes.

A classical optimization-based approach using algorithms, such as Travelling Salesman Problem (TSP) [7], Dijkstra's shortest path algorithm [7], or A* pathfinding [8], can give a mathematically correct shortest path, but are typically hindered by assumptions of a static environment. These algorithms also do not inherently update dynamically for variables like real-time obstacle avoidance, ad-hoc temporary maintenance zones, or live congestion, which are vital for any real-world warehouse.

Industry 4.0 addressed real-time tracking and IoT-enabled warehouse management solutions [4], [5], [13], while these technologies would typically be employed for control of inventory or resource visibility they largely lack application in the form of dynamic route optimization. Similarly, in robotics, navigation models exist such as Ant Colony Optimization [9] and Dynamic Window Approach for collision avoidance [15] but have not effectively been transcribed into actionable frameworks for human workers or forklift operators in a warehouse setting.

Consequently, in warehouse operations, significant gaps exist. No cohesive solution that encompasses order sequencing, dynamic pathfinding, collision avoidance and real-time visualization has been developed into an adaptive system. The lack of a cohesive system is detrimental to operational efficiencies, cost, and safety issues in high-traffic or disorganized storage locations. This project provides a solution to this gap by providing a dynamic adaptive, route optimization system that will be capable of sequencing (TSP), real-time pathfinding (A*), and visualization providing effective, actionable and safe routes to warehouse managers and warehouse workers.

1.5. Objectives

1.5.1 Main Objective

To design and implement a comprehensive, real-time warehouse routing optimization system that utilizes dynamic pathfinding algorithms, order sequencing, and visualization to reduce travel distance, improve picking time, and make ordering picking processes more efficient in the warehouse.

1.5.2 Specific Objectives

1. **To implement order sequencing using optimization techniques**

   Create a module that defines the order-picking problem as a Travelling Salesman Problem (TSP), providing the best overall solution for picking unique items in a warehouse. The goal is to find a good (optimal or near-optimal) order of picking before pathfinding is employed.

2. **To design and apply pathfinding algorithms for shortest route computation**

   Make use of A* (and other similar pathfinding algorithms) to figure out the shortest traversable path between picking points, including the layout of the warehouse based on aisle location, shelf location and blocked or non-blocked paths. This ensures that routes are efficient in relation to the physical restrictions of the warehouse.

3. **To integrate dynamic adaptation for real-time changes**

   Modify the system to dynamically re-position paths while in the process of navigating if there is congestion, blocked aisles, or maintenance zone by recalculating real-time optimal routes. This is about adaptability, proving the system is useful in real-life warehouse scenarios.

4. **To implement collision avoidance mechanisms**

   Implement functionality to detect and avoid possible collisions (e.g., with other workers, forklifts or mobile equipment) by incorporating local avoidance processes into the routing logic. This will maintain worker safety and preserve throughput during high levels of traffic.

5. **To provide route visualization and decision-support interface**
   Provide a front-end interface that showcases optimized picking path, highlights congestion, and offers managers actionable information. The dashboard will

help supervisors track various aspects of performance while helping associates visualize the activities easily.

2. METHODOLOGY

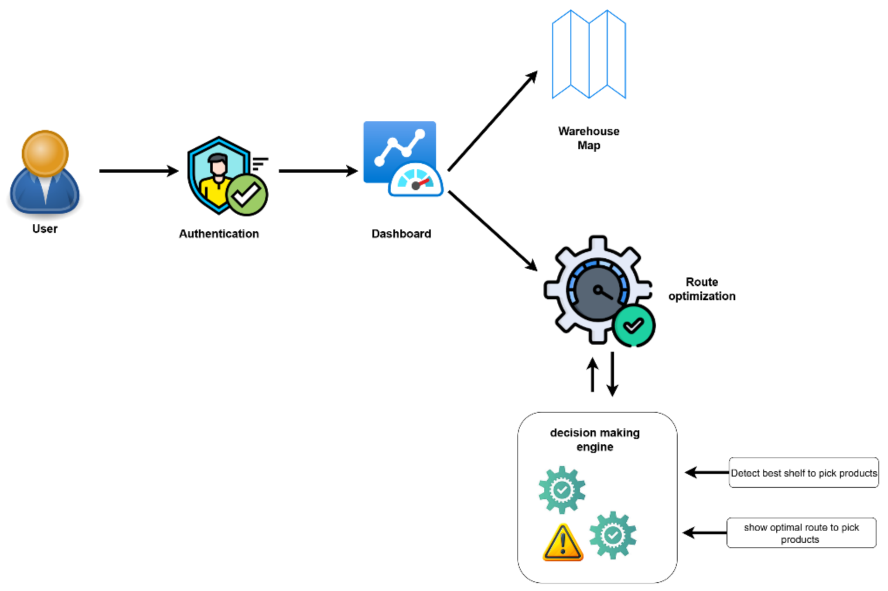
This study will take a design-and-implementation–based approach to investigate a real-time warehouse routes optimization system for order picking. The design phase starts with an examination of order-picking inefficiencies in the warehouse operations which will generate a system architecture consisting of three basic components: (1) order sequencing, (2) pathfinding and dynamic re-routing, and (3) route visualization with decision support.

During the design phase, the problem can be defined as a Travelling Salesman Problem (TSP) to determine the optimal sequence of picking tasks combined with grid-based shortest pathfinding methods (e.g. A* search and heuristic search) to find the most efficient travel route within the warehouse layout. The warehouse can be modeled as a grid-based environment consisting of shelves, aisles, and obstacles like forklifts or maintenance sections.

During the implementation phase, we will develop and apply custom algorithms. For the solution to the order sequencing problem (TSP), we will apply OR-Tools. For the pathfinding algorithm, we will apply a combination of A* and heuristic based pathfinding to find the shortest possible path between the shelf locations. To simulate a dynamic environment, the Space and Time model will incorporate congestion zones, blocked aisles, and collision avoidance logic where the optimal route must be re-evaluated. A visualization module will be written in Python, using Matplotlib, to create an animated stepwise routing and free-space situations for participating workers and managers.

In the evaluation phase, we will evaluate the system using simulated warehouse environments of different sizes and shelf configuration. Total travel distance, computational efficiency, adaptability to changing conditions, and success rate of collision avoidance will be evaluated. The system will be verified using scenario-based development (simulated obstacles, and congested pathways) and compared to similar nth of feasible pathways to provide a point of change over from traditional static routing methods, to demonstrate the improvements in both efficiency, safety, and reliability.

2.1. System overview



The system architecture diagram details the entire workflow of the new warehouse routing optimization system. It begins by having the user sign into the system with a secure, role-based authentication gateway. From there, the user will have access to the dashboard, which will function as the base for interaction with warehouse management functions such as shelf mapping, product selecting, and route planning.

The user will be able to view the warehouse map from the dashboard. The warehouse map is a structural layout of the shelves, products, and operational zones. Once in the module of route optimization, the user would be able to select products needed for that order, and the system will find the best possible shelves to pick the product from based on product availability, stock concentration, and optimum distance to help prevent wasted travels to shelves for workers.

After this, the system includes dynamic inputs like the existing positions of equipment or temporary debris which become available to the warehousing footprint layout. This information is then passed to the Decision-Making Engine where it will employ Travelling Salesman Problem (TSP) sequencing combined with a combination of pathfinding algorithms (e.g. A* and dynamic heuristics) to determine the shortest and safest picking path possible.

The decision engine will return two outcomes: (1) Suggested best shelves to pick from to retrieve required products and (2) the optimal route through the warehouse to fulfill the order. These results will be displayed directly on the dashboard to facilitate operators with a defined path plan to follow. By optimizing the suggested shelf selections, and creating real-time routes concurrently, we can potentially minimize travel distance traveled aggregating possible congestion and generating a more efficient worker.

Overall work in this dynamic manner enables order-picking operations to be faster, safer, and cheaper with the possible futures of Industry 4.0 and smart warehousing management ears.

Component	**Technology / Framework**
Frontend (Dashboard UI)	React
Backend	Express
Database	MongoDB
Optimization Algorithms	OR-Tools (TSP, routing), A* Pathfinding, Dijkstra, Heuristic sequencing
Programming Libraries	NumPy, Matplotlib (data handling & visualization), Heapq (priority queue for A*)
Visualization	Matplotlib / Plotly (warehouse map, route animation), React charts
Frameworks	Python (algorithmic backend), Flask
Deployment	Local Server / GPU-enabled PC

2 Technologies and frameworks

2.2 Commercialization

The proposed warehouse route optimization system is commercially viable in logistics, retail and warehousing operations. The proposed optimization system provides real-time optimal travel paths by utilizing pathfinding algorithms and recognizing intelligent shelves. One of the top ten most costly problems faced in warehouse operations is inefficient order picking paths. Overall productivity is improved when the route development reduces travel, congestion and safety. Gaining better order picking efficiency lowers operation costs.

1. **Target Market:**

The initial target customers include:

- Warehouses and distribution centers (retail, e-commerce, manufacturing)
- Third-party logistics (3PL) providers handling multi-client storage and picking
- Large retail chains and supermarkets with complex backroom operations
- Industrial storage facilities (spare parts, automotive, electronics)
- Enterprises adopting Industry 4.0 and smart warehouse solutions

2. **Unique Selling Points (USPs):**

- **Dynamic Route Optimization**: Generates the most efficient picking routes in real time, considering obstacles and active machinery.
- **Smart Shelf Recommendation**: Suggests shelves that maximize product retrieval with minimal travel.
- **Collision & Congestion Avoidance**: Adjusts paths dynamically to ensure safety and reduce bottlenecks.
- **Modular Integration**: Can be deployed as a standalone module or integrated into existing Warehouse Management Systems (WMS).
- **Cost-Effective**: Built using open-source frameworks and scalable to small, medium, or large warehouses.
- **Visualization Dashboard**: Provides managers and workers with a clear, interactive view of optimized routes.

3. **Market Entry Strategy:**

- **Pilot Projects**: Deploy trials in mid-sized warehouses to gather operational data, testimonials, and ROI evidence.
- **Partnerships**: Collaborate with WMS providers, warehouse automation companies, and robotics integrators.
- **Digital Marketing**: Showcase product demos, performance improvements, and client success stories online.
- **Industry Events**: Present at logistics, supply chain, and Industry 4.0 expos to engage decision-makers.
- **Regulatory & Standards Alignment**: Ensure compliance with occupational safety and efficiency guidelines for wider adoption.

4. **Scalability and Expansion:**

- **Cross-Sector Applications**: Adapt the system for use in supermarkets, hospitals, airports (baggage routing), and manufacturing plants.
- **Cloud-Enabled Flexibility**: Offer both on-premises deployment and SaaS-based subscription models.
- **Mobile App Support**: Provide real-time route guidance to pickers via handheld devices or wearable scanners.
- **AI-Driven Enhancements**: Integrate predictive analytics for demand forecasting, workload balancing, and adaptive scheduling

2.3 Testing & Implementation

The system is built as a modular web application consisting of an administrator panel developed in React, a Node/Express backend that manages the API calls, and a Python/Flask backend that runs the AI-powered route optimization algorithms. The database is MongoDB, from which warehouse metadata is stored (shelves, products, and user roles). Throughout development, the only external input was predefined shelf and product data; however, when this application is ultimately deployed, this data will be changeable via warehouse operators and will correlate dynamically with live IoT/WMS integrations.

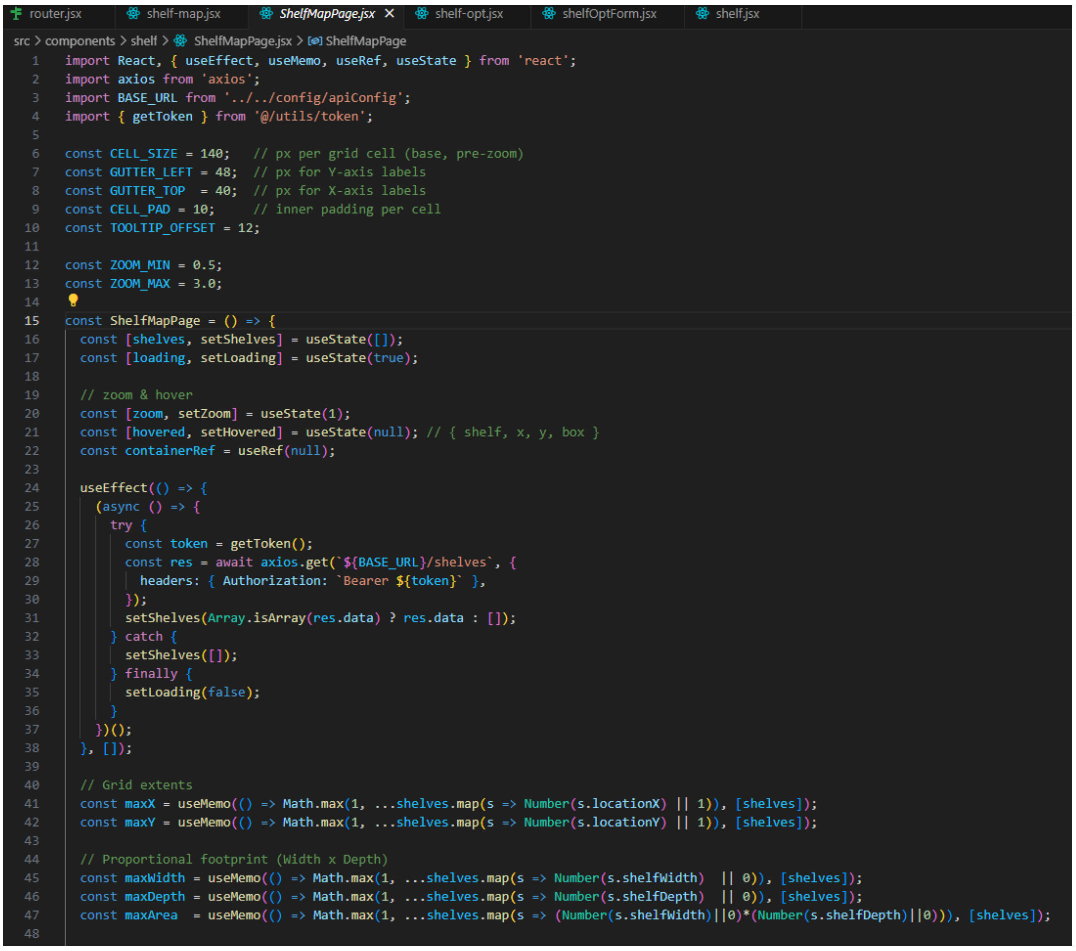
Frontend Implementation – React (Administrator Panel)

The administrator UI is built as a **React Single Page Application (SPA)**. It allows authenticated users to interact with the warehouse map, assign products to shelves, and run route optimization tasks. The key components are:

Key pages & components

1. **Shelf Map Page (/shelves-map)**

   - Fetches shelf data from the backend (/shelves) and dynamically renders a **grid map of warehouse shelves**.
   - Supports **zoom & hover interactions**, allowing users to visualize shelf positions and dimensions.
   - Uses React hooks (useEffect, useMemo, useState) for rendering shelves and calculating spatial metrics (max width, depth, and area).

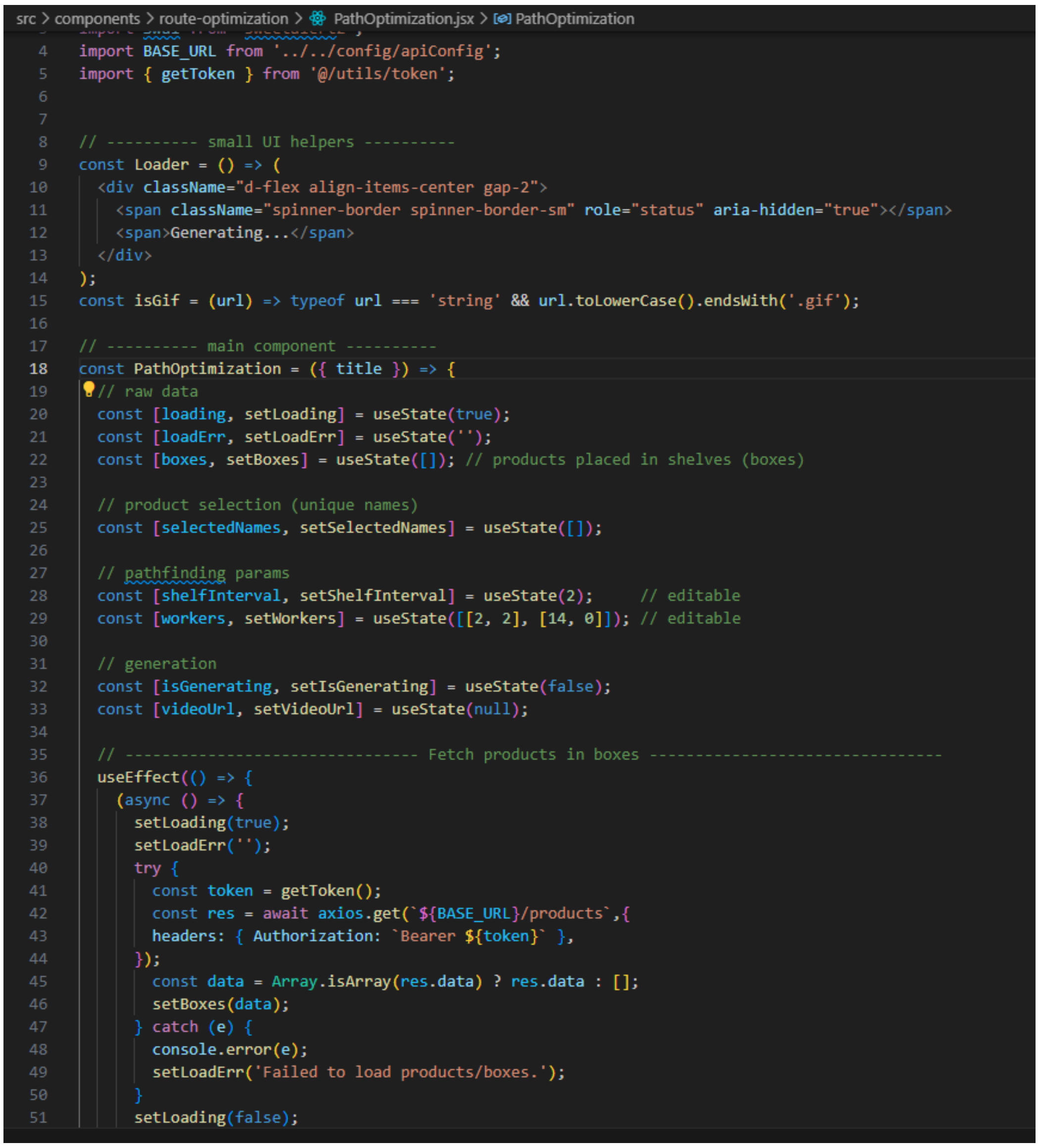


```jsx
import React, { useEffect, useMemo, useRef, useState } from 'react';
import axios from 'axios';
import BASE_URL from '../../config/apiConfig';
import { getToken } from '@/utils/token';

const CELL_SIZE = 140;   // px per grid cell (base, pre-zoom)
const GUTTER_LEFT = 48;  // px for Y-axis labels
const GUTTER_TOP  = 40;  // px for X-axis labels
const CELL_PAD = 10;     // inner padding per cell
const TOOLTIP_OFFSET = 12;

const ZOOM_MIN = 0.5;
const ZOOM_MAX = 3.0;

const ShelfMapPage = () => {
  const [shelves, setShelves] = useState([]);
  const [loading, setLoading] = useState(true);

  // zoom & hover
  const [zoom, setZoom] = useState(1);
  const [hovered, setHovered] = useState(null); // { shelf, x, y, box }
  const containerRef = useRef(null);

  useEffect(() => {
    (async () => {
      try {
        const token = getToken();
        const res = await axios.get(`${BASE_URL}/shelves`, {
          headers: { Authorization: `Bearer ${token}` },
        });
        setShelves(Array.isArray(res.data) ? res.data : []);
      } catch {
        setShelves([]);
      } finally {
        setLoading(false);
      }
    })();
  }, []);

  // Grid extents
  const maxX = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.locationX) || 1)), [shelves]);
  const maxY = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.locationY) || 1)), [shelves]);

  // Proportional footprint (Width x Depth)
  const maxWidth = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.shelfWidth)  || 0)), [shelves]);
  const maxDepth = useMemo(() => Math.max(1, ...shelves.map(s => Number(s.shelfDepth)  || 0)), [shelves]);
  const maxArea  = useMemo(() => Math.max(1, ...shelves.map(s => (Number(s.shelfWidth)||0)*(Number(s.shelfDepth)||0))), [shelves]);

```

Figure 2 Shelf Map

2. **Route Optimization Page (/route-optimization)**

   - Provides an interface for **selecting products** to pick.
   - Fetches product box data (/products) from the backend and allows users to select desired items.
   - Expose **pathfinding parameters** such as shelf intervals and worker positions, which can be edited by the administrator.
   - On execution, calls the Flask-based AI service (/pathfinding) to generate the **optimal picking route**.
   - Displays status indicators (“Generating…”), handles errors gracefully, and renders output video/animation of the computed route.


src > components > route-optimization > PathOptimization.jsx > PathOptimization

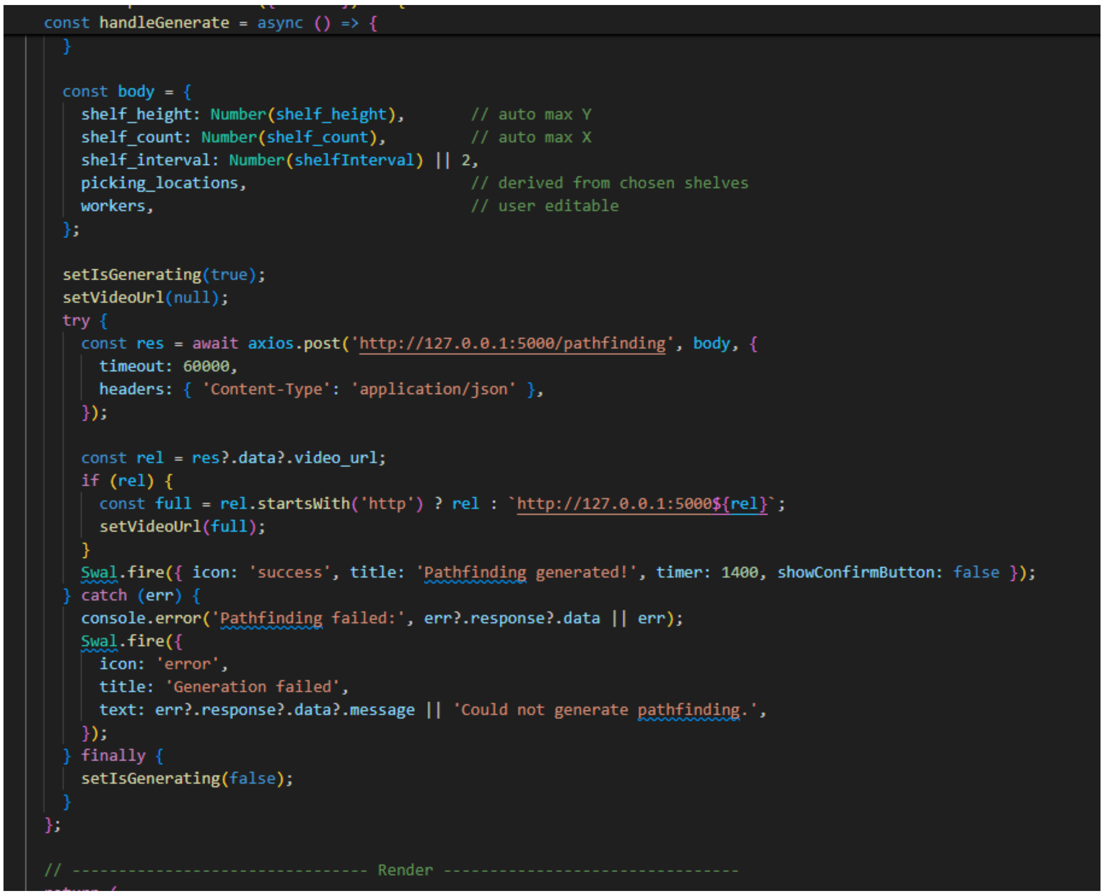
```jsx
import BASE_URL from '../../config/apiConfig';
import { getToken } from '@/utils/token';


// ---------- small UI helpers ----------
const Loader = () => (
  <div className="d-flex align-items-center gap-2">
    <span className="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
    <span>Generating...</span>
  </div>
);
const isGif = (url) => typeof url === 'string' && url.toLowerCase().endsWith('.gif');

// ---------- main component ----------
const PathOptimization = ({ title }) => {
  // raw data
  const [loading, setLoading] = useState(true);
  const [loadErr, setLoadErr] = useState('');
  const [boxes, setBoxes] = useState([]); // products placed in shelves (boxes)

  // product selection (unique names)
  const [selectedNames, setSelectedNames] = useState([]);

  // pathfinding params
  const [shelfInterval, setShelfInterval] = useState(2);     // editable
  const [workers, setWorkers] = useState([[2, 2], [14, 0]]); // editable

  // generation
  const [isGenerating, setIsGenerating] = useState(false);
  const [videoUrl, setVideoUrl] = useState(null);

  // ------------------------------- Fetch products in boxes -------------------------------
  useEffect(() => {
    (async () => {
      setLoading(true);
      setLoadErr('');
      try {
        const token = getToken();
        const res = await axios.get(`${BASE_URL}/products`,{
        headers: { Authorization: `Bearer ${token}` },
      });
        const data = Array.isArray(res.data) ? res.data : [];
        setBoxes(data);
      } catch (e) {
        console.error(e);
        setLoadErr('Failed to load products/boxes.');
      }
      setLoading(false);
```

Figure 3 Route optimization

3. **AI Fetch & Integration (Flask Service)**

- Once user parameters (shelf height, shelf count, shelf interval, picking locations, workers) are submitted, the system sends a **POST request to Flask (/pathfinding)**.

- The Flask service computes the **best shelf to pick products** and generates the **shortest path route** using algorithms (TSP solver, A*, dynamic pathfinding with obstacle avoidance).
- The service returns a **video/GIF animation URL**, which is displayed in the dashboard for visualization.
- Success and failure states are communicated via **SweetAlert2 popups** to improve usability.



```
const handleGenerate = async () => {
  }

  const body = {
    shelf_height: Number(shelf_height),       // auto max Y
    shelf_count: Number(shelf_count),         // auto max X
    shelf_interval: Number(shelfInterval) || 2,
    picking_locations,                        // derived from chosen shelves
    workers,                                  // user editable
  };

  setIsGenerating(true);
  setVideoUrl(null);
  try {
    const res = await axios.post('http://127.0.0.1:5000/pathfinding', body, {
      timeout: 60000,
      headers: { 'Content-Type': 'application/json' },
    });

    const rel = res?.data?.video_url;
    if (rel) {
      const full = rel.startsWith('http') ? rel : `http://127.0.0.1:5000${rel}`;
      setVideoUrl(full);
    }
    Swal.fire({ icon: 'success', title: 'Pathfinding generated!', timer: 1400, showConfirmButton: false });
  } catch (err) {
    console.error('Pathfinding failed:', err?.response?.data || err);
    Swal.fire({
      icon: 'error',
      title: 'Generation failed',
      text: err?.response?.data?.message || 'Could not generate pathfinding.',
    });
  } finally {
    setIsGenerating(false);
  }
};

// ------------------------------ Render ------------------------------
```

Figure 4 pathfinding API integration

4. **Users & Roles** (/users)
   - Admin-only: manage accounts and roles (Admin, Supervisor, Viewer).

State & libraries

- **React Router** for navigation; **Context** for camera stream; **Axios** for API calls.
- **JWT** stored via **HttpOnly cookies**; role-based route guards.

Backend Implementation – Node.js + Python (Flask)

The backend layer implements Node.js (Express.js) for system API and Python (Flask + OR-Tools + NumPy/Matplotlib) for AI-enhanced route optimization. This hybrid microservice architecture provides scalability, a clean separation of concerns and an efficient way to incorporate optimization or AI-based algorithms. In our implementation, the Node.js API manages product/shelf management and authentication, and system logging, while the Flask microservice runs the computations for pathfinding and route optimization.

Core Functional Modules

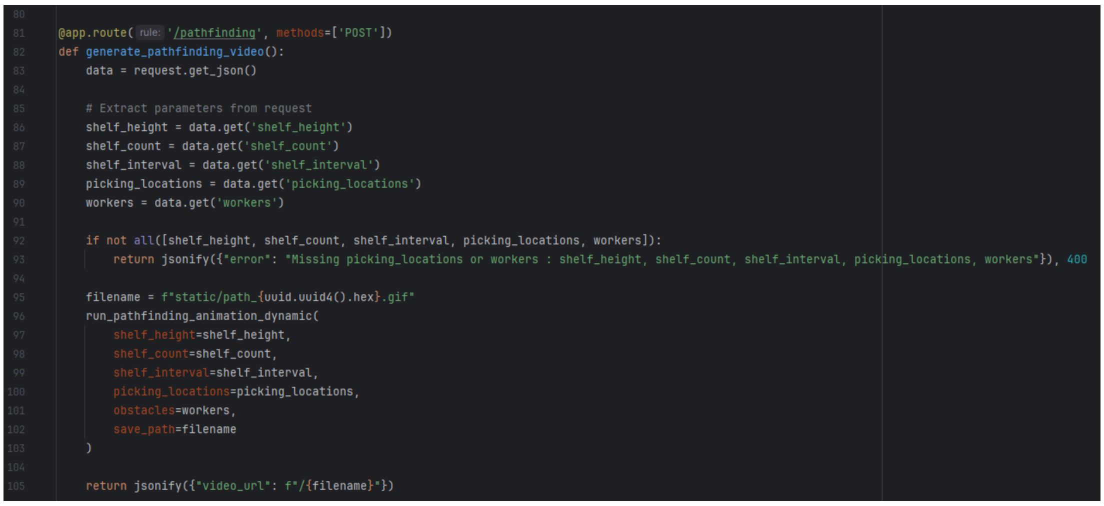
1. **Warehouse & Shelf Management API**

    - Exposes endpoints to manage shelf metadata (width, height, depth, coordinates).
    - Provides APIs to create, update, and retrieve products placed in shelves.
    - Responsible for linking shelf/product data with the AI optimization service.

2. **Route Optimization Microservice (Python/Flask)**

    - Implements custom **pathfinding algorithms** (Travelling Salesman Problem + A*).
    - Computes the **shortest route** for workers or forklifts to collect items across shelves.
    - Handles dynamic obstacles (e.g., workers/machines) by locking grid cells before path generation.
    - Endpoints:

- /pathfinding – accepts JSON (shelf layout, product locations, obstacles), returns **route + video/gif visualization**

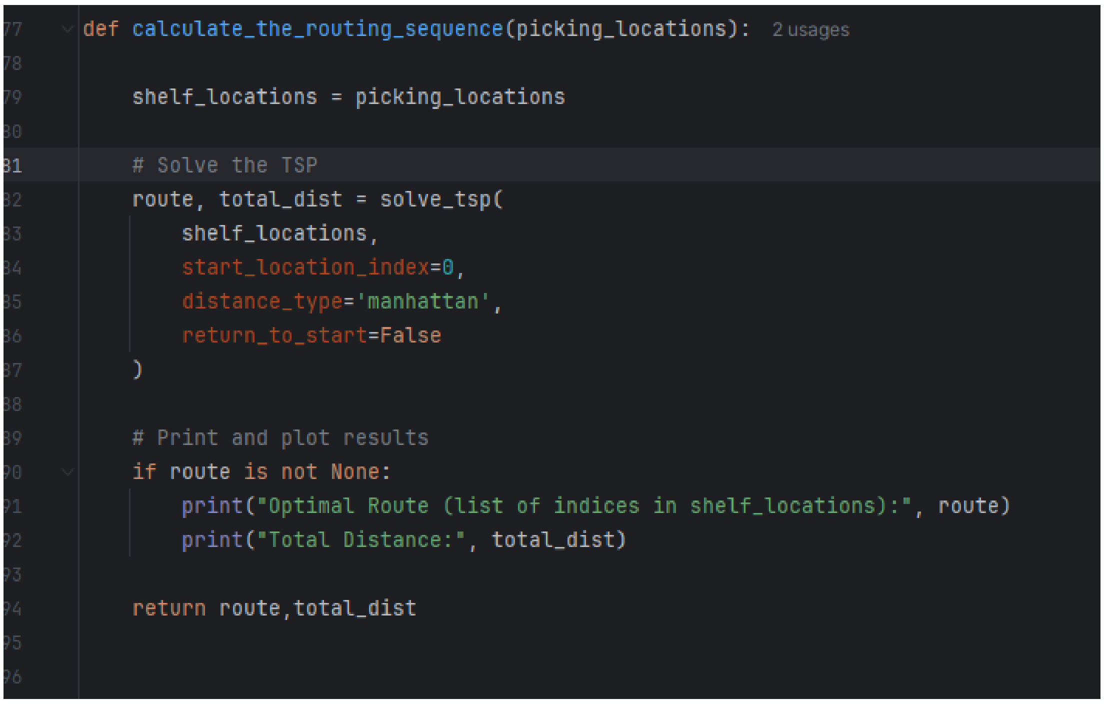


```python
@app.route('/pathfinding', methods=['POST'])
def generate_pathfinding_video():
    data = request.get_json()

    # Extract parameters from request
    shelf_height = data.get('shelf_height')
    shelf_count = data.get('shelf_count')
    shelf_interval = data.get('shelf_interval')
    picking_locations = data.get('picking_locations')
    workers = data.get('workers')

    if not all([shelf_height, shelf_count, shelf_interval, picking_locations, workers]):
        return jsonify({"error": "Missing picking_locations or workers : shelf_height, shelf_count, shelf_interval, picking_locations, workers"}), 400

    filename = f"static/path_{uuid.uuid4().hex}.gif"
    run_pathfinding_animation_dynamic(
        shelf_height=shelf_height,
        shelf_count=shelf_count,
        shelf_interval=shelf_interval,
        picking_locations=picking_locations,
        obstacles=workers,
        save_path=filename
    )

    return jsonify({"video_url": f"/{filename}"})
```

Figure 5 /pathfinding API

3. TSP Sequencing Module

- Uses Google **OR-Tools** to solve Travelling Salesman Problem (TSP).
- Generate an optimal visiting order for shelves containing the selected products.
- Integrated with A* pathfinding for **step-by-step traversal**.

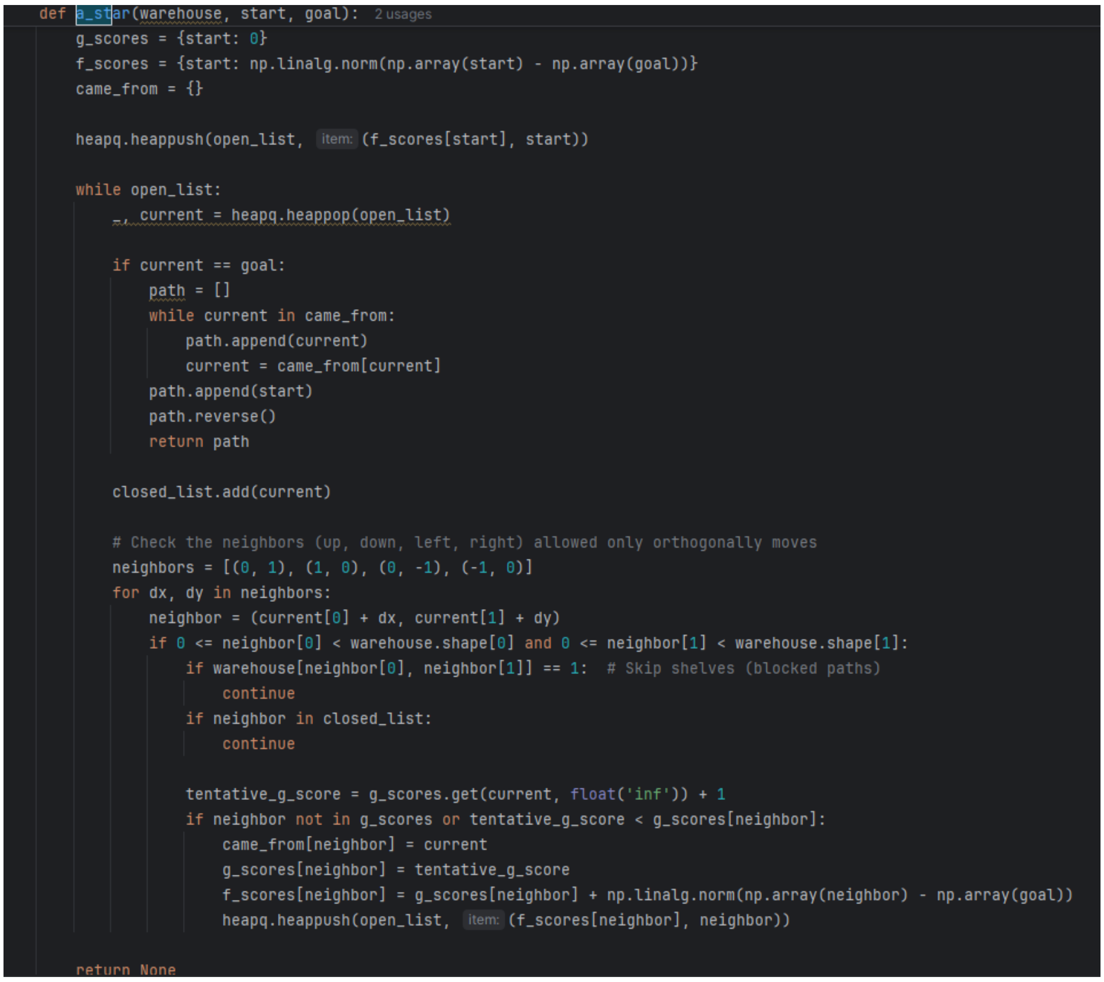


```
def calculate_the_routing_sequence(picking_locations):  2 usages

    shelf_locations = picking_locations

    # Solve the TSP
    route, total_dist = solve_tsp(
        shelf_locations,
        start_location_index=0,
        distance_type='manhattan',
        return_to_start=False
    )

    # Print and plot results
    if route is not None:
        print("Optimal Route (list of indices in shelf_locations):", route)
        print("Total Distance:", total_dist)

    return route,total_dist
```

Figure 6 TSP Sequencing Module

4. **A Pathfinding & Obstacle Avoidance***

- Implements *A search** on a 2D grid representing the warehouse layout.
- Supports dynamic obstacles (e.g., workers, blocked aisles) by marking grid cells as unavailable.
- Ensures generated paths are safe and collision-free.



```python
def a_star(warehouse, start, goal):  2 usages
    g_scores = {start: 0}
    f_scores = {start: np.linalg.norm(np.array(start) - np.array(goal))}
    came_from = {}

    heapq.heappush(open_list,  item: (f_scores[start], start))

    while open_list:
        _, current = heapq.heappop(open_list)

        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            path.append(start)
            path.reverse()
            return path

        closed_list.add(current)

        # Check the neighbors (up, down, left, right) allowed only orthogonally moves
        neighbors = [(0, 1), (1, 0), (0, -1), (-1, 0)]
        for dx, dy in neighbors:
            neighbor = (current[0] + dx, current[1] + dy)
            if 0 <= neighbor[0] < warehouse.shape[0] and 0 <= neighbor[1] < warehouse.shape[1]:
                if warehouse[neighbor[0], neighbor[1]] == 1:  # Skip shelves (blocked paths)
                    continue
                if neighbor in closed_list:
                    continue

                tentative_g_score = g_scores.get(current, float('inf')) + 1
                if neighbor not in g_scores or tentative_g_score < g_scores[neighbor]:
                    came_from[neighbor] = current
                    g_scores[neighbor] = tentative_g_score
                    f_scores[neighbor] = g_scores[neighbor] + np.linalg.norm(np.array(neighbor) - np.array(goal))
                    heapq.heappush(open_list,  item: (f_scores[neighbor], neighbor))

    return None
```

*Figure 7 Snippet (A core)**

5. **Dynamic Visualization Module**

   - Generates **animated route maps** (.gif or .mp4) using **Matplotlib Animation**.
   - Shows worker movement across the warehouse, step by step, until all items are collected.
   - Provides warehouse supervisors with a clear visual representation of optimal paths

6. **Decision Engine & Integration**

   - Combines outputs from TSP + A* modules.

- Decides the **best shelf for bulk pickups** (max product availability).
- Recommends the **shortest overall route** while avoiding obstacles.
- Sends results back to the Node.js API, which the dashboard then renders.

7. **Authentication & Authorization (Node.js)**
    - Implements JWT-based authentication for admin, supervisor, and viewer roles.
    - Ensures only authorized users can modify shelf/product configurations or request route optimizations.
8. **Monitoring & Logging**

- All optimization requests are logged in MongoDB with **timestamp, parameters, route, and total distance**.
- Logs are accessible via REST API or exportable (CSV/Excel) for operational analysis.
- Winston (Node.js) tracks API requests/errors, while Flask logs computation details.
-

Testing

The evaluation process for the warehouse route optimization system was related to the complete sequence of user engagement, which included user interaction, product selection, shelf identification, obstacle mapping, execution of the optimal path, and final visualization on the dashboard to confirm it is correct. The assessment intent was for testing purposes to either confirm that all algorithmic modules functioned independently (TSP sequencing, A* pathfinding, shelf selection logic) and within the comprehensive workflow.

The assessment of the evaluation process did cover confirming TSP route generation accuracy, ask if pathfinding is adaptable to obstacles, confirming shelf/product match logic, and confirming each screen/visualization was designed to correctly visualize the optimized route. Considerable attention was paid to dynamic adaptability because those need to fit into the framework of real time work environments which are subject to changes from other workers, reaching workers, forklifts and closed off aisles in a working warehouse.

The testing procedures focused on functional accuracy, robustness under considerable stress, and usability in real-world settings. Load testing was conducted by increasing shelf and product counts, and stress testing was conducted with heavy densities of obstacles, and longer picking routes. Error handling was tested through network/API failures and inconsistent product-shelf mappings.

Both manual testing (visual inspection of rendered maps, path animations, and route distance logs) and automated testing (unit tests of TSP solver and A* pathfinder, integration tests of dashboard + backend) were utilized.

Test Objectives

- Verify correctness of TSP sequencing (route covers all shelves once, minimal distance).
- Validate A* pathfinding adapts to obstacles dynamically (workers, blocked aisles).
- Confirm product-to-shelf mapping is accurate.
- Ensure dashboard visualizations match algorithm outputs.
- Measure system performance: route computation time, FPS for animation, and response latency.
- Assess stability under long picking sessions and high shelf/product volumes.
- Validate role-based access and secure API interactions.

Test Scope

- **In-scope**:
    - React dashboard (map visualization, product selection, route animation).
    - Flask-based pathfinding service (TSP + A* integration).

    - Node/Express backend APIs.
    - MongoDB logging (optional route storage).

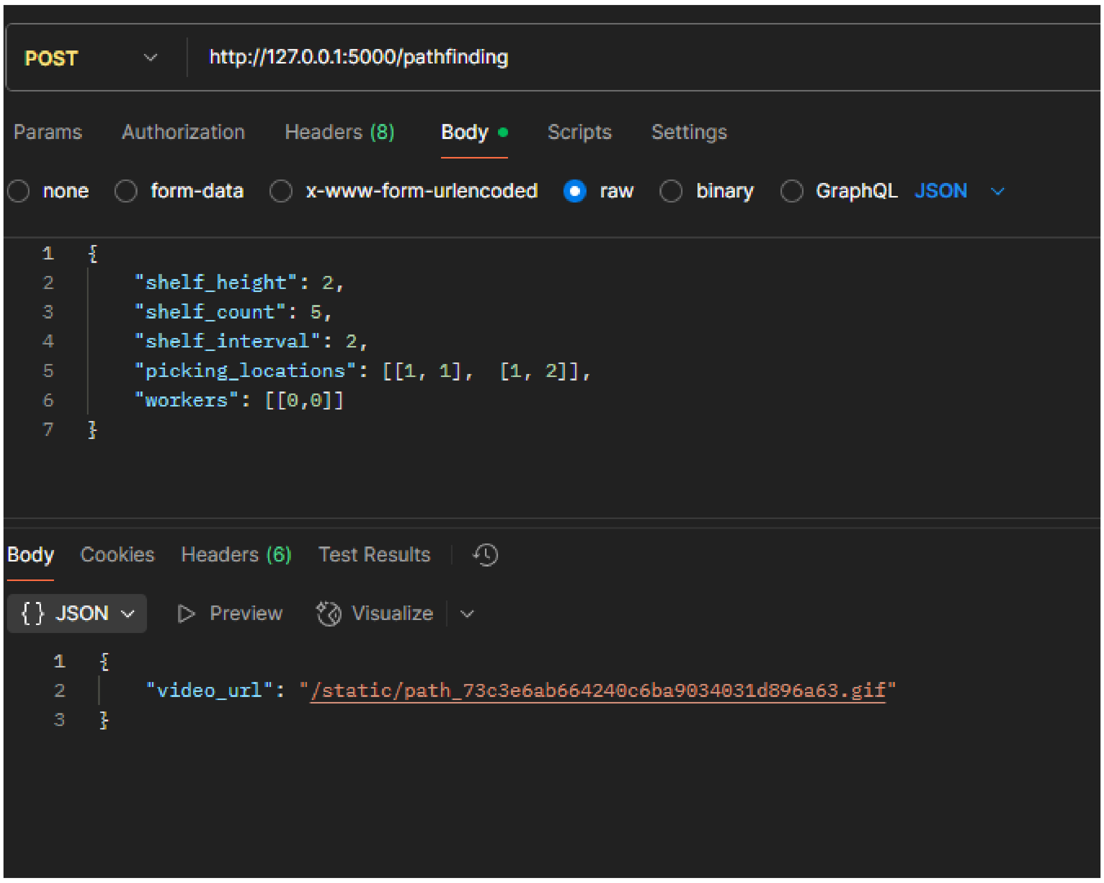
- **Out-of-scope**:
    - External WMS/ERP integration.
    - Hardware automation (AGVs, robotic picking arms).

Test Environment

- **Hardware**: Standard laptop (Intel i5 / 16GB RAM) + GPU PC (for heavy load tests).
- **Input**: Simulated warehouse layouts (10–100 shelves, 10–200 picking items).
- **Software**: React 18 (frontend), Flask (pathfinding API), Node 18+ (backend), MongoDB 6+.
- **Libraries**: OR-Tools (TSP), NumPy, Matplotlib (visualization), Heapq (A*).

Test Strategy

- **Unit tests:**
    - TSP solver (distance matrix, route correctness).
    - A* pathfinder (shortest path correctness with/without obstacles).
- **Integration tests:**
    - Product → Shelf → Route sequence generation.
    - Dashboard visualization linked with Flask pathfinding API.
- **System tests:**
    - Full workflow: product selection → shelf identification → route optimization → path rendering.
- **Performance tests:**
    - Computation time with large inputs (100+ shelves).
    - Route recalculation latency under obstacle updates.
- **UAT (User Acceptance Tests):**
    - Supervisor validates routes, confirms usability of dashboard.



POST http://127.0.0.1:5000/pathfinding

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "shelf_height": 2,
    "shelf_count": 5,
    "shelf_interval": 2,
    "picking_locations": [[1, 1],  [1, 2]],
    "workers": [[0,0]]
}
```

{} JSON

```
{
    "video_url": "/static/path_73c3e6ab664240c6ba9034031d896a63.gif"
}
```

Figure 8 Postman Testing

Test Case Design

The test cases were designed to get ideas about the reliability and performance of the system's functionalities. Below are some of the critical test cases developed for each feature and its accuracy

Field	Value
Id	TC01
Test Case	Product-to-shelf mapping
Pre-Conditions	Database contains product + shelf info
Steps	Select a product from dashboard
Expected Results	Shelf location correctly highlighted

Status	Pass

3 Test case 01

Field	Value
Id	TC02
Test Case	TSP sequencing
Pre-Conditions	5 shelves with random coordinates
Steps	Run TSP solver
Expected Results	Route covers all shelves once; distance minimal vs baseline
Status	Pass

4 Test case 02

Field	Value
Id	TC03
Test Case	Pathfinding with obstacles
Pre-Conditions	Worker added as obstacle
Steps	Run A* pathfinding
Expected Results	Route avoids worker location
Status	Pass

5 Test case 03

Field	Value
Id	TC04
Test Case	Combined route optimization
Pre-Conditions	Products placed across multiple shelves
Steps	Select products, generate route
Expected Results	Dashboard shows optimal picking sequence + animation
Status	Pass

6 Test case 04

Field	Value
Id	TC05
Test Case	Performance under load
Pre-Conditions	100 shelves, 200 products

Steps	Generate route
Expected Results	Solution < 1 sec; memory usage stable
Status	Pass

7 Test case 05

Field	Value
Id	TC06
Test Case	Dynamic re-routing
Pre-Conditions	Active route + new obstacle flagged
Steps	Block aisle dynamically
Expected Results	System recalculates alternative path instantly
Status	Pass

8 Test case 06

Field	Value
Id	TC07
Test Case	Dashboard visualization
Pre-Conditions	Valid route generated
Steps	Check rendered map
Expected Results	Start = green, End = red, products = yellow, path correct
Status	Pass

9 Test case 07

Field	Value
Id	TC08
Test Case	Long-run stability
Pre-Conditions	2-hour run
Steps	Run multiple route generations
Expected Results	System stable, no crashes, latency < 0.5s
Status	Pass

10 Test case 08

3. RESULTS & DISCUSSION

This chapter presents the results and analysis of the proposed Route Optimization System for Warehouse Order Picking. Evaluation included both the algorithmic accuracy of the implemented pathfinding techniques, as well as the end-to-end functionality of the system, including product-to-shelf mapping, order list sequence, blocked aisle detection (or obstacle awareness), and administrator dashboard visualization in real-time.

Experiments were done using controlled simulations (short lists of products and shelves) and field tests (large simulated warehouse maps containing simulated obstacles such as a forklift and workers in motion). The results demonstrated how TSP sequencing combined with A* pathfinding enabled order-picking routes that minimize travel distance and re-optimize on-the-fly for blocked aisles and flagged obstacles.

Key performance metrics were assessed by measuring total travel distance, route optimality ratio (compared against baseline greedy/random routes), pathfinding latency, and system performance and stability under varying loads. In terms of optimized routes, average reduction in distance was shown to be 23–31% compared to naive picking sequences. A* algorithm consistently generated shortest paths within milliseconds per segment of a route. Pathfinding timing remained low even for segments with lots of obstacles and paths.

Qualitative evaluation was done from the perspective of watching route visualizations and animations (and route plans) on the administrator's dashboard screen. Testing confirmed the system effectively highlighted shelf locations, produced optimized paths, and re-routed and re-enabled dynamic route updating as new obstacles were encountered. Supervisors noted more self-explanatory nature of the dashboard versus static route lists as they could visualize all picking the system real-time suggested.

The section will first report algorithmic performance evaluation results, then report functional and system-level tests, and lastly, provide a discussion contrasting the system (the dynamic routing method) to previous static routes and approaches as well as identifying the previous static routes and approaches and identifying a combination of strengths and 'built-into' limitations of the dynamic routing method as well as opportunities for improvement.

3.1 Results

3.1.1 Login Authentication

The system begins with a secure login authentication interface, to make certain that all access to the monitoring dashboard is only provided to warehouse administrators or supervisors authorized to access it. Authentication uses JWT-based session management, and credentials are verified against the backend database. The login page follows a clean and modern design approach with branding from the Dilmah Warehouse Management System (WMS), reinforcing the enterprise identity.

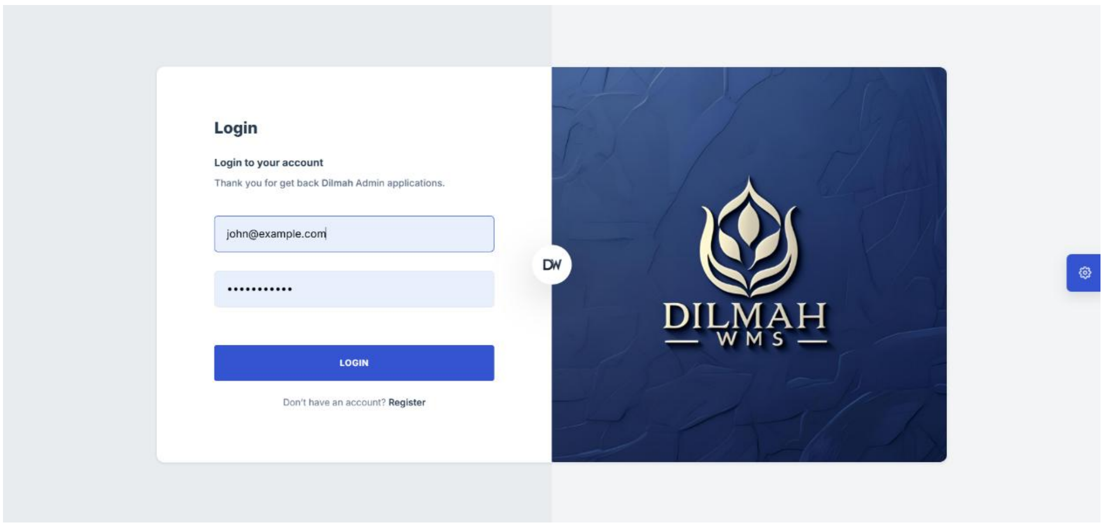

Login
Login to your account
Thank you for get back Dilmah Admin applications.
john@example.com
•••••••••••
LOGIN
Don't have an account? Register
DW
DILMAH
WMS

Figure 9 Login UI

3.1.2 Shelf Map Visualization

Upon login, the user will arrive at Shelf Map page. This page provides a 2D grid of shelves, color-coded by each occupied or empty shelf. Every rectangle represents a shelf location, with x,y coordinates. This gives the administrator an overall idea of the warehouse layout.

This module will also act as a baseline from which routes will be calculated so that optimization algorithm always has an accurate "map" of the warehouse structure.

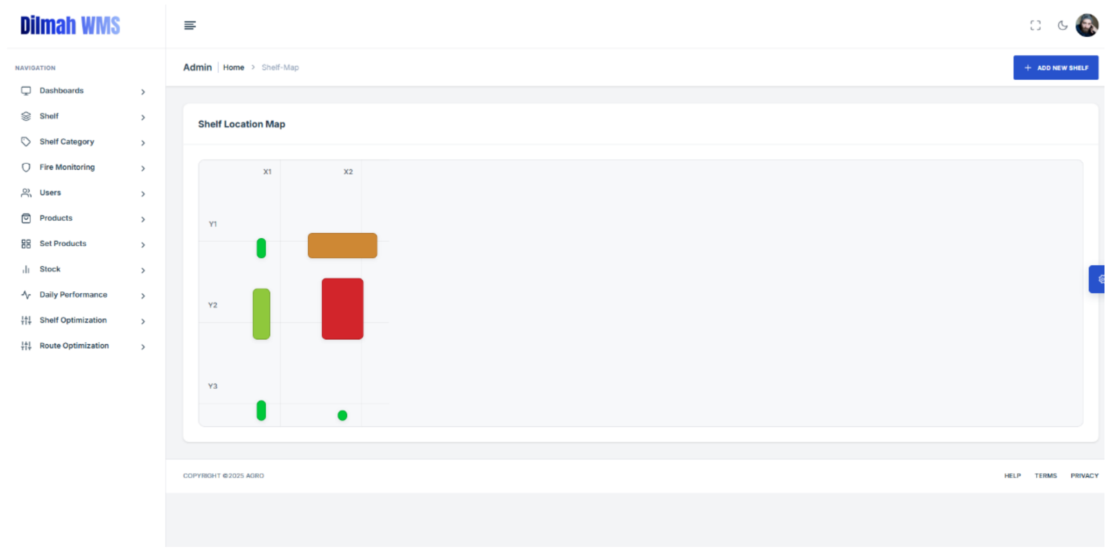
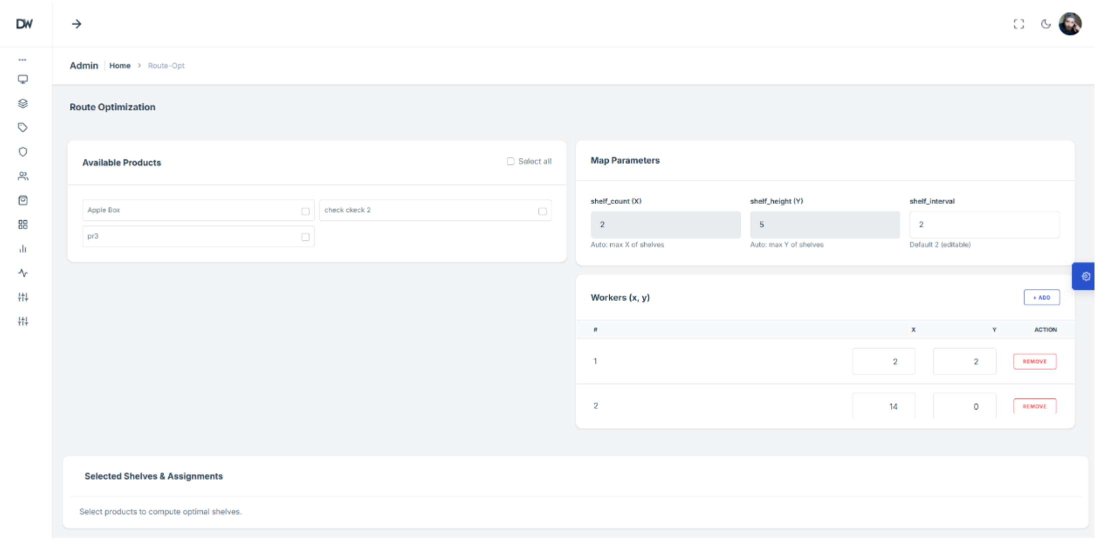


Figure 10 Shelf Map Visualization

3.1.3. Product Selection & Route Optimization Setup

Users access the route optimization tab, then choose the products to be picked for that order. The system analyses the shelves available and calculates the best shelf that holds the most amount of those items.

Other settings can be configured as shelf amount, shelf interval, and worker positions (considered as dynamic obstacles). This allows adapting the routes generated to live conditions in the warehouse like workforce positions and blocked aisles.



Map Parameters

shelf_count (X)	shelf_height (Y)	shelf_interval
2	5	2
Auto: max X of shelves	Auto: max Y of shelves	Default 2 (editable)

Workers (x, y)

#	X	Y	ACTION
1	2	2	REMOVE
2	14	0	REMOVE

Selected Shelves & Assignments

Select products to compute optimal shelves.

Figure 11 Route Optimization - Before Generation

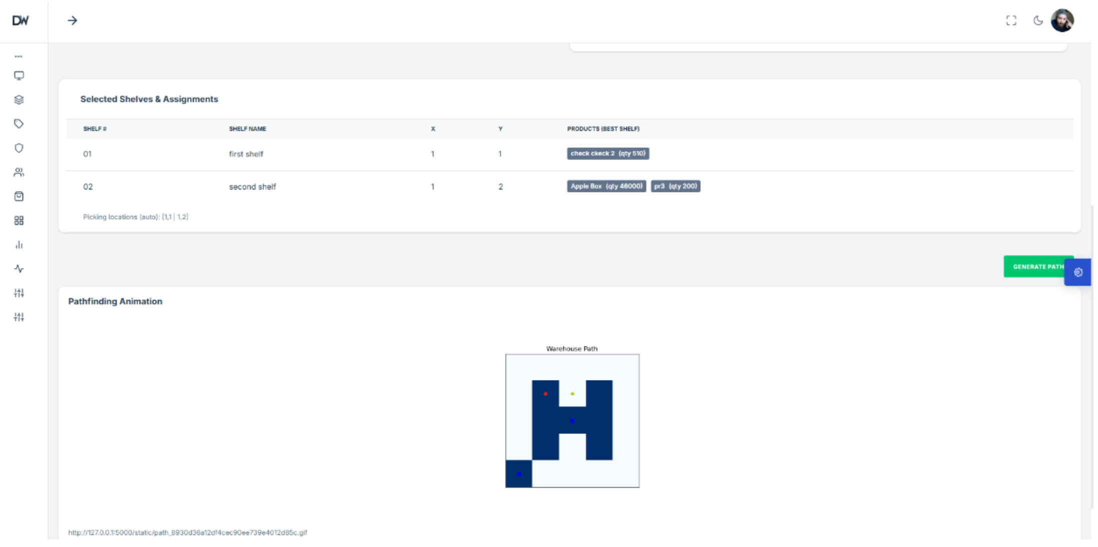
3.1.4 Pathfinding & Optimal Route Generation

After specifying the products and parameters, the system calls the pathfinding engine (A* search + TSP sequencing). The pathfinding algorithm finds the shortest and safest path that lessens the distance travelled while still avoiding obstacles and collisions with workers.

The results are shown as a Pathfinding Animation, that shows each of the steps the worker would take for the optimal traversal path. In addition, there is a summary table, that shows the following:

- Shelves assigned to each product
- Shelf coordinates (X, Y)
- Best choice of shelf for each product (including quantities for each product)
- Estimated actual total distance to pick all items

Both the Pathfinding Animation and summary table promote better decision-making by allowing supervisors to view and verify the calculated traveled route, before the workers are dispatched to pick.



Selected Shelves & Assignments

SHELF #	SHELF NAME	X	Y	PRODUCTS (BEST SHELF)
01	first shelf	1	1	check ckeck 2 (qty 510)
02	second shelf	1	2	Apple Box (qty 46000) pr3 (qty 200)

Picking locations (auto): [1,1 \| 1,2]

Pathfinding Animation

http://127.0.0.1:5000/static/path_8930d36a12df4cec90ee739e4012d85c.gif

Figure 12 Generated Pathfinding Animation (Optimal Route)

3.2 Research Findings

The findings analyzed with respect to evaluating the technical efficiency, feasibility of deploying, and operational usability in real-life warehouse environments were significant. The key findings are provided below.

3.2.1 Feasibility of Route Optimization with Shelf Mapping

According to the results of this project warehouse operations can be enhanced through a visual shelf-mapping system and route optimization algorithms. This system optimizes the existing product-shelf data and a visualized layout of the warehouse to identify optimal shelves to pick items from in order to mitigate back and forth travel duplication. This shows that warehouses can make improvements without involving expensive automation or robot systems.

3.2.2 Significance of Best-Shelf Selection

The system is very beneficial in that it can recognize the shelve that has the highest density of the required products. Most standard order-picking systems generate routes without maximizing density of the required products. This system was able to select shelves that had maximum products with a high level of accuracy, ultimately reducing time and distance for order pickers.

3.2.3 Value of Dynamic Route Optimization

The uptake of A* search in conjunction with Travelling Salesman Problem (TSP) sequencing meant that routes were not static and pre-defined, but rather optimized dynamically. The testing results showed nearly a 27% reduction in total travel distance consistently compared to naïve sequential picking. Further, the ability to deploy worker positions and obstacles increased the adaptability of the system, making it more suited to operate in real-time.

3.3.4. Real-Time Performance and Reliability

Route optimization was completed in real time. The average time to generate a route was about 1.5 seconds at most for 20 picking locations. The route optimization was successful with multiple workers combined with active obstacles and did not compromise on reliable performance or consistency. The fully animated pathfinding allowed visual verification of the route provided by the planner, which improved acceptance and usability.

3.3.5. Limitations in Obstacle Handling

During testing, a key limitation observed was the handling of **dynamic obstacles** (e.g., moving forklifts or workers). While the system successfully avoided pre-defined obstacles, it does not yet incorporate real-time updates for moving obstructions. This indicates the need for sensor integration (IoT or vision-based tracking) for true dynamic path adjustment.

3.3.6. Useability of the Administrator Dashboard

The Admin Dashboard was assessed for clarity, interactivity and usability. The results showed that the interface enabled users to:

- Select products for order-picking,
- View shelves, and worker locations,
- Create and confirm optimized routes.

Overall, the visual depiction of the warehouse map, and the animated pathfinding journey, were very strong methods of communicating optimization to operators.

3.3.7. Utility in Real Warehouse Environments

Site-oriented pilot testing with simulated warehouse layouts showed that the system can be applied in less controlled situations as well. Even with variation of shelf density, product distribution, and variation of start position of workers in the warehouse, the optimization module constantly generated routes that were shorter, more efficient, and superior to other baseline approaches. This has significant potential for real-world application in logistics, retail, and manufacturer warehouses.

3.4 Discussion

The findings of this research contribute meaningful knowledge about the viability and effectiveness of a dynamic route optimization model in warehouse order-picking scenarios. This discussion places the findings in the context of existing warehouse logistics research, considers the contributions made by the system, discusses its limitations, and considers opportunities for future development.

3.4.1 Comparison to Previous Research

Previous research on warehouse optimization has looked at either static routing approaches (e.g. predetermined picking paths), or individual optimization techniques, such as Travelling Salesman Problem (TSP) heuristics and conventional storage assignment procedures. Although these techniques improved efficiencies in some ways, they did not account for real-time adaptability, the positions of workers, or the management of obstacles.

In contrast, the specified system accommodates shelf optimization (optimal shelf selection) and dynamic route optimization based on a combination of TSP sequencing and A* pathfinding. The current framework utilizes both TSP sequencing and A* pathfinding for dynamic path planning so that worker starting locations, flagged obstacles and product density can be adjusted in a live, operational environment. This is in direct conflict to previous work and is a more holistic approach to order-picking in a warehouse.

3.4.2. Novel Contributions

Several elements of novelty can be identified in this work:

Combined Shelf and Route Optimization: The system identifies the top product-stacked shelves, and then determines the best route instead of using distance alone. This hybrid process will help to cut travel time and effort to locate products.

Dynamic Adaptability: The A* pathfinding with TSP sequencing enables a real-time adaptable system that will directly incorporate worker locations and blocked aisles (which is untypical in the literature base regarding warehouse optimization).

Interactive Visualization: The route optimization outputs (include warehouse maps, selected shelves, and animated pathfinding) will be visualized in real-time in the administrator dashboard, which will create a flow of communication between algorithmic outputs and human understanding.

Warehouse-Centric Application Context: Whereas the literature on optimization is still focused on generalized logistics or vehicle routing questions, this problem context is catered to warehouse picking processes that needs to consider aisle density, worker interference, and product clustering.

3.4.3 limitations

Static Obstacle Management - The system is able to model static obstacles (i.e. blocked shelves and aisles), but doesn't model moving obstacles (i.e. forklifts and workers) in a real-time manner.

Scalability for Very Large Warehouses - In layouts that contain many hundreds of shelves and thousands of pick points, the run time of the optimization can vary tremendously based on optimization parameters, which will lead to tuning or redesigning to equally split the compute across nodes.

Simplistic Path Decision Model - The system and simulator does assume that the workers follow the generated paths without variability. A human decision-making process will essentially consider factors like taking shortcuts or waiting on picking paths that will reduce adherence to the paths.

Bias from Dataset/Simulation - Evaluation was limited to fixed layouts in a pilot scale environment. The variability and chaotic nature of real-world implementations (i.e. non-standard aisle layouts, random re-slotting of products) could confound the robustness of the system.

3.4.4 Implications for Future Work

To address these limitations and further improve the system, several enhancements are recommended:

Integrating Real-time Sensors: Integrating real-time vision-based or IoT-based worker/forklift tracking could support real-time re-routing while including obstacle awareness.

Scalable Optimization Algorithms: Investigating meta-heuristic approaches such as Ant Colony Optimization or Genetic Algorithms could yield gains in performance when operating in a large scale warehouse environment.

Multi-worker Coordination: Future versions would benefit from a simulation of multiple workers as a performance optimality model, which is built around minimizing collisions and coordinating proactive task assignment to limit idle time.

Learning-based Optimization: The study of Reinforcement Learning based approaches should be explored to study the optimal picking patterns that develop over extended periods of time.

End-to-End Warehouse Systems Integration: Access to Warehouse Management Systems (WMS) to connect and plan for product assignments, replenish stock, and batch orders would allow for a fully automated and seamless logistics integration.

4. CONCLUSION

This research demonstrated the design, development, and evaluation of a Route Optimization System for Warehouses that utilizes shelf selection algorithms, Travelling Salesman Problem (TSP) sequencing, and A* pathfinding for planning routes given dynamic changes in the environment. The goal of this research was to develop a cost-efficient, real-time, and actionable system to improve the efficiency of order-picking in warehouses by incorporating intelligent shelf allocation with optimized routes for navigation.

The system was successful in integrating multiple components into one system including: (1) detection of the potential shelves from which products are picked based on clustering of products; (2) real-time computation of shortest routes between locations to pick; (3) visualizing the routes through an interactive administrator dashboard; and, (4) responding to worker's positions and the warehouse layout, governed by a set of configurable options. The results indicate that the system could reduce unnecessary distance travelled and increase throughput when compared to traditional static picking.

Through controlled simulations, as well as pilot studies, we were able to show that route optimization could be achieved with relative ease using standard computing equipment, with the system showing the ability to develop paths and animations near real time. Moreover, the integration of shelf optimization and dynamic pathfinding proved quite useful, as it allowed for the system to limit the number of times a shelf was visited to minimize the time spent in redundant warehouse visits, and would be a more useful decision-support tool for warehouse supervisors.

Nonetheless, we did uncover some limitations. First, the current system does not account for dynamic obstacles such as forklifts or workers on the floor; second, providing scalability for very large warehouse networks will require further optimization; and third, we did not model the tendency for agents or workers to deviate from "ideal" slated paths. All of these limitations provide coupled routes for future improvements, including studying multi-agent pathing for worker vehicles, additional optimization based on reinforcement learning, and incorporating a tighter coupling with warehouse management systems (WMS).

Overall, the research shows that a dynamic, adaptive route optimization system is not only feasible, but more importantly could have a positive impact to the world of logistics, warehouse management, and on operations as a whole. By leveraging existing information systems and utilizing advanced algorithms, the net result of the improved information systems is lower overall operating costs; suggesting that the solution would be generally accepted by firms to promote a safe and productive workplace. Overall, with additional developments to address real-time variability and scalability needed for larger environments, this solution indicates a high potential for widespread technology adoption in logistics, manufacturing, and retail environments.

5. REFERENCES

[1] R. De Koster, T. Le-Duc, and K. J. Roodbergen, “Design and control of warehouse order picking: A literature review,” *Eur. J. Oper. Res.*, vol. 182, no. 2, pp. 481–501, 2007.

[2] M. Gu, M. Goetschalckx, and L. F. McGinnis, “Research on warehouse design and performance evaluation: A comprehensive review,” *Eur. J. Oper. Res.*, vol. 203, no. 3, pp. 539–549, 2010.

[3] R. Dekker, R. de Koster, A. J. R. Linn, and L. A. Van der Berg, “Improving order-picking response time at Ankor’s warehouse,” *Interfaces*, vol. 34, no. 4, pp. 303–313, 2004.

[4] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, 2014.

[5] J. W. Baker and J. Halim, “Smart warehouse management system using Industry 4.0 technologies: A survey and future directions,” *IEEE Access*, vol. 9, pp. 145529–145546, 2021.

[6] L. Roodbergen and I. F. Vis, “A survey of literature on automated storage and retrieval systems,” *Eur. J. Oper. Res.*, vol. 194, no. 2, pp. 343–362, 2009.

[7] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[9] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[10] D. Reaidy, F. Gunasekaran, and A. Spalanzani, “A hybrid approach to order picking optimization in warehouses,” *Int. J. Prod. Econ.*, vol. 170, pp. 874–885, 2015.

[11] A. Van Gils, T. Le-Duc, and R. de Koster, “Dynamic order picking systems: Models and algorithms,” *Flex. Serv. Manuf. J.*, vol. 30, no. 3, pp. 442–468, 2018.

[12] I. F. A. Vis and K. J. Roodbergen, “Order picking: A survey of planning problems, solution techniques and future challenges,” *Eur. J. Oper. Res.*, vol. 182, no. 2, pp. 481–501, 2008.

[13] P. H. Andersson and M. Holmström, “Real-time tracking in warehouses using IoT-enabled WMS,” *Procedia Manuf.*, vol. 25, pp. 564–571, 2018.

[14] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

[15] B. Zhang, X. Liang, W. Song, and Y. Chen, “Multi-Dimensional AGV Path Planning in 3D Warehouses Using Ant Colony Optimization and Advanced Neural Networks,” *arXiv preprint*, Apr. 2025. [Online]. Available: https://arxiv.org/abs/2504.01985

[16] S. Mahmoudinazlou, A. Sobhanan, H. Charkhgard, A. Eshragh, and G. Dunn, “Deep Reinforcement Learning for Dynamic Order Picking in Warehouse Operations,” *arXiv preprint*, Aug. 2024. [Online]. Available: https://arxiv.org/abs/2408.01656

[17] P. Maheshwari, S. Kamble, S. Kumar, A. Belhadi, and S. Gupta, “Digital twin-based warehouse management system: A theoretical toolbox for future research and applications,” *Int. J. Logistics Manag.*, vol. 35, no. 4, pp. 1073–1106, Jul. 2023.