Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2019, 6(11):325-332**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Shipping Estimation

Praveen Kumar Vutukuri

WMS – Warehouse Management System (of Affiliation)
Walgreens (of Affiliation)
California, USA
praveen524svec@gmail.com

Abstract: This paper explores the methods and technologies used in shipping estimation and the online order fulfillment process at Walgreens. It examines the process of creating shipping estimates, the use of carrier APIs, and the specific strategies employed by Walgreens to handle online orders efficiently. This implementation is completely related to the warehouse application and once the online order was placed by a customer, how does online order data come through the warehouse system and how does fulfillment happen.

Keywords: Shipping estimation, Order fulfillment, E-commerce logistics, Carrier APIs, Delivery estimation, Inventory management, Automated shipping, Retail logistics, Walgreens, Customer satisfaction, Real-time shipping rates, Supply chain optimization, Shipping zones, Handling fees, Delivery options, Shipping software, Order processing, Online retail, Carrier integration.

Introduction

In the age of e-commerce, accurate shipping estimation and efficient order fulfillment are critical to customer satisfaction and operational success. This paper delves into the methodology of creating shipping estimates and provides a case study on Walgreens, a major retail pharmacy chain, to illustrate how these processes are implemented in practice.

Shipping Estimation Methodology

Shipping estimation involves multiple steps to ensure precision and reliability. The following outlines the general methodology:

A. Gather Required Information

1. Package Details: Dimensions, weight, and contents.
2. Origin and Destination: Addresses or zip codes for the sender and recipient.
3. Shipping Method: Standard, express, overnight, etc.
4. Carrier Options: Available shipping carriers (e.g., UPS, FedEx, DHL, USPS).

```
using System;
using System.Collections.Generic;

namespace ShippingEstimation
{
    public class ShippingInfo
    {
        public PackageDetails Package { get; set; }
        public Address Origin { get; set; }
        public Address Destination { get; set; }
        public ShippingMethod Method { get; set; }
        public List<CarrierOption> Carriers { get; set; }

        public ShippingInfo()
        {
            Carriers = new List<CarrierOption>();
        }
    }

    public class PackageDetails
    {
        public Dimensions Dimensions { get; set; }
        public double Weight { get; set; }
        public string Contents { get; set; }
    }

    public class Dimensions
    {
        public double Length { get; set; }
        public double Width { get; set; }
        public double Height { get; set; }
    }

    public class Address
    {
        public string Street { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string ZipCode { get; set; }
        public string Country { get; set; }
    }

    public enum ShippingMethod
    {
        Standard,
        Express,
        Overnight
    }
```

As part of the above example for each UPC (Universal Product Code) will be mapped with the respective details before creating shipping estimation and execute through below process.

B. Calculate Package Weight and Dimensions

Accurate weight and dimensions are critical. Using precise scales and measuring tools, dimensions are converted to the carrier's preferred unit.

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace ShippingEstimationAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ShippingController : ControllerBase
    {
        [HttpPost("convert")]
        public ActionResult<ConvertedPackageDetails> ConvertPackageDetails([FromBody] PackageDetails packageDetails)
        {
            if (packageDetails == null)
            {
                return BadRequest("Invalid package details.");
            }

            var convertedDimensions = ConvertDimensionsToPreferredUnit(packageDetails.Dimensions);
            var convertedWeight = ConvertWeightToPreferredUnit(packageDetails.Weight);

            var convertedPackageDetails = new ConvertedPackageDetails
            {
                Dimensions = convertedDimensions,
                Weight = convertedWeight
            };

            return Ok(convertedPackageDetails);
        }

        private Dimensions ConvertDimensionsToPreferredUnit(Dimensions dimensions)
        {
            // Assuming the carrier prefers centimeters for dimensions
            const double conversionFactor = 2.54; // inches to centimeters

            return new Dimensions
            {
                Length = dimensions.Length * conversionFactor,
                Width = dimensions.Width * conversionFactor,
                Height = dimensions.Height * conversionFactor
            };
        }

        private double ConvertWeightToPreferredUnit(double weight)
        {
            // Assuming the carrier prefers kilograms for weight
            const double conversionFactor = 0.453592; // pounds to kilograms

            return weight * conversionFactor;
        }
    }
```

```
public class PackageDetails
{
    public Dimensions Dimensions { get; set; }
    public double Weight { get; set; }
    public string Contents { get; set; }
}

public class Dimensions
{
    public double Length { get; set; }
    public double Width { get; set; }
    public double Height { get; set; }
}

public class ConvertedPackageDetails
{
    public Dimensions Dimensions { get; set; }
    public double Weight { get; set; }
}
}
```

As shown in the above code each UPC's details converted to the format as per Carrier's preference. Each carrier's preference will be saved in the database tables as one time save.

C. Select a Shipping Carrier

Choosing the right carrier involves comparing rates, reliability, and delivery times. Each carrier has a unique pricing model.

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;

namespace ShippingEstimationAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ShippingController : ControllerBase
    {
        [HttpPost("chooseCarrier")]
        public ActionResult<CarrierOption> ChooseBestCarrier([FromBody] ShippingRequest request)
        {
            if (request == null || request.Carriers == null || !request.Carriers.Any())
            {
                return BadRequest("Invalid request or no carriers provided.");
            }

            var bestCarrier = CompareCarriers(request.Package, request.Carriers);

            return Ok(bestCarrier);
        }

        private CarrierOption CompareCarriers(PackageDetails package, List<CarrierOption> carriers)
        {
            // Dummy logic for comparison, in a real-world application,
            // this would involve calling each carrier's API to get real-time rates, reliability, and delivery times
            return carriers.OrderBy(carrier => carrier.Price)
                           .ThenBy(carrier => carrier.Reliability)
                           .ThenBy(carrier => carrier.EstimatedDeliveryTime)
                           .FirstOrDefault();
        }
    }

    public class ShippingRequest
    {
        public PackageDetails Package { get; set; }
        public List<CarrierOption> Carriers { get; set; }
    }

    public class PackageDetails
    {
        public Dimensions Dimensions { get; set; }
        public double Weight { get; set; }
        public string Contents { get; set; }
    }
```

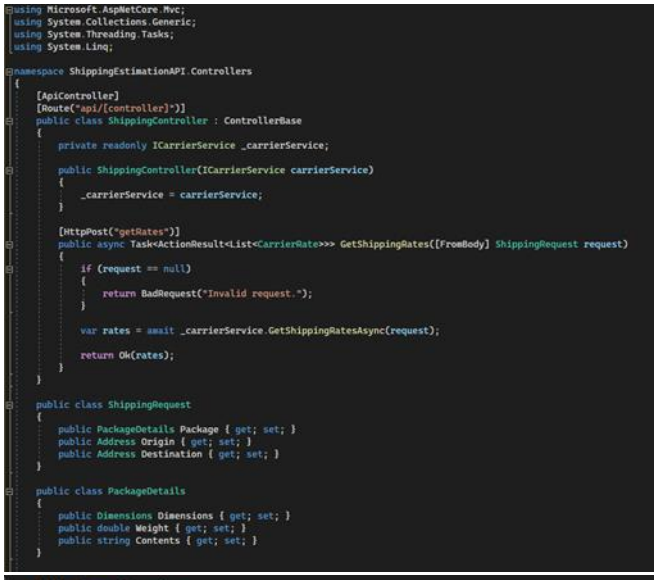
```
public class Dimensions
{
    public double Length { get; set; }
    public double Width { get; set; }
    public double Height { get; set; }
}

public class CarrierOption
{
    public string Name { get; set; }
    public double Price { get; set; } // In USD
    public double Reliability { get; set; } // A score from 0 to 1
    public int EstimatedDeliveryTime { get; set; } // In days
    public string ServiceCode { get; set; }
}
}
```

As per the above sample logic business module will sort out all the carriers by reliability, price and delivery time. As we are going to estimate and choose the better carrier for shipment.

D. Use Carrier Rate Tables or APIs

Carriers provide rate tables or APIs for real-time shipping costs, accounting for factors like fuel surcharges and insurance.



```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Linq;

namespace ShippingEstimationAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ShippingController : ControllerBase
    {
        private readonly ICarrierService _carrierService;

        public ShippingController(ICarrierService carrierService)
        {
            _carrierService = carrierService;
        }

        [HttpPost("getRates")]
        public async Task<ActionResult<List<CarrierRate>>> GetShippingRates([FromBody] ShippingRequest request)
        {
            if (request == null)
            {
                return BadRequest("Invalid request.");
            }

            var rates = await _carrierService.GetShippingRatesAsync(request);

            return Ok(rates);
        }
    }

    public class ShippingRequest
    {
        public PackageDetails Package { get; set; }
        public Address Origin { get; set; }
        public Address Destination { get; set; }
    }

    public class PackageDetails
    {
        public Dimensions Dimensions { get; set; }
        public double Weight { get; set; }
        public string Contents { get; set; }
    }
```

```
    public class Dimensions
    {
        public double Length { get; set; }
        public double Width { get; set; }
        public double Height { get; set; }
    }

    public class Address
    {
        public string Street { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string ZipCode { get; set; }
        public string Country { get; set; }
    }

    public class CarrierRate
    {
        public string CarrierName { get; set; }
        public double Rate { get; set; }
        public double FuelSurcharge { get; set; }
        public double Insurance { get; set; }
        public double TotalCost { get; set; }
    }

    public interface ICarrierService
    {
        Task<List<CarrierRate>> GetShippingRatesAsync(ShippingRequest request);
    }

    public class CarrierService : ICarrierService
    {
        public async Task<List<CarrierRate>> GetShippingRatesAsync(ShippingRequest request)
        {
            // Simulating API calls to different carriers
            var rates = new List<CarrierRate>
            {
                new CarrierRate
                {
                    CarrierName = "UPS",
                    Rate = 20.00,
                    FuelSurcharge = 5.00,
                    Insurance = 2.00,
                    TotalCost = 27.00
                },
```

```
                new CarrierRate
                {
                    CarrierName = "FedEx",
                    Rate = 18.00,
                    FuelSurcharge = 4.50,
                    Insurance = 2.50,
                    TotalCost = 25.00
                }
                // Add more carriers as needed
            };

            // In a real-world application, this method would make asynchronous API calls to the carrier services.

            await Task.Delay(100); // Simulating async call delay

            return rates;
        }
    }
}
```

In the above piece of code, you can refer as some sample code details we are getting from carrier's API with all the details.

E. Determine Shipping Zones

Carriers use geographic zones to determine rates. The zone for a shipment is found using the carrier's zone chart.

```
public async Task<List<CarrierRate>> GetShippingRatesWithZonesAsync(ShippingRequest request)
{
    var originZip = request.Origin.ZipCode;
    var destinationZip = request.Destination.ZipCode;

    var zone = FindZone(originZip, destinationZip);
    var rates = GetRatesForZone(zone);

    await Task.Delay(100); // Simulating async call delay

    return rates.Select(rate => new CarrierRate
    {
        CarrierName = rate.CarrierName,
        Rate = rate.Rate,
        FuelSurcharge = rate.FuelSurcharge,
        Insurance = rate.Insurance,
        TotalCost = rate.TotalCost,
        Zone = zone
    }).ToList();
}

private int FindZone(string originZip, string destinationZip)
{
    if (_zoneChart.ContainsKey(originZip) && _zoneChart[originZip].ContainsKey(destinationZip))
    {
        return _zoneChart[originZip][destinationZip];
    }

    // Default zone if not found
    return 1;
}
```

```
var rates = new List<CarrierRate>
{
    new CarrierRate
    {
        CarrierName = "UPS",
        Rate = 20.00 + zone,
        FuelSurcharge = 5.00,
        Insurance = 2.00,
        TotalCost = 27.00 + zone
    },
    new CarrierRate
    {
        CarrierName = "FedEx",
        Rate = 18.00 + zone,
        FuelSurcharge = 4.50,
        Insurance = 2.50,
        TotalCost = 25.00 + zone
    }
    // Add more carriers as needed
};
```

As default zone data would be saved on organization's db and as part of the cost calculation these will be calculated as Miscellaneous charges.

F. Estimate Delivery Time

Delivery times vary based on the shipping method and carrier, accounting for weekends and holidays.

Delivery Times: Factors Influencing Variability

1. Shipping Method

• Standard Shipping:

O Delivery Time: Typically ranges from 3 to 7 business days, depending on the distance between the origin and destination.

O Details: Standard shipping is often the most economical but also the slowest. It may involve several hand-offs and longer transit times through regional hubs.

• Express Shipping:

O Delivery Time: Usually 1 to 3 business days.

O Details: Express services prioritize speed and often involve expedited processing and handling. Carriers such as FedEx, UPS, and DHL offer express options like overnight or 2-day delivery.

• Overnight Shipping:

O Delivery Time: Guaranteed delivery by the next business day.

O Details: This method ensures the fastest delivery, often with a premium cost. It's commonly used for urgent or high-value items.

• Two-Day Shipping:

O Delivery Time: Guaranteed delivery within 2 business days.

O Details: Often used for items that need to be delivered quickly but do not require overnight service. Many e-commerce platforms offer this as a standard option for premium customers.

2. Carrier Differences

• FedEx:

O Service Types: Includes options such as FedEx Express (overnight and 2-day delivery) and FedEx Ground (economical delivery with longer transit times).

O Transit Times: FedEx provides guaranteed delivery times for express services but varies for ground services depending on distance and location.

• UPS:

O Service Types: Offers UPS Next Day Air, UPS 2nd Day Air, and UPS Ground.

O Transit Times: UPS provides precise delivery windows for express services and estimated delivery times for ground services based on zone charts.

• DHL:

O Service Types: Known for international shipping, including DHL Express for fast global delivery.

O Transit Times: DHL Express provides time-sensitive delivery options with guaranteed delivery times, especially useful for international shipments.

• USPS:

O Service Types: Includes Priority Mail (1-3 day delivery) and First-Class Mail (1-5 day delivery for packages).

O Transit Times: USPS delivery times vary based on the service used and the distance between the origin and destination.

3. Weekends and Holidays

• Weekends:

O Impact: Most carriers do not count weekends as business days. For example, a shipment with a 2-day express delivery service may not be delivered on weekends if shipped on a Friday.

O Exceptions: Some carriers, like FedEx and UPS, offer Saturday delivery options for certain services, often with additional fees.

• Holidays:

O Impact: National holidays and public holidays can affect delivery times. Carriers may not operate or have reduced operations on these days, causing delays.

O Planning: During holiday seasons (e.g., Christmas, Thanksgiving), increased shipping volumes can also impact delivery times, potentially extending estimated delivery windows.

Practical Implications

• Businesses must account for these variations when providing delivery estimates to customers. Understanding the specific transit times for each shipping method and carrier allows for more accurate delivery promises.

• Clear communication about expected delivery times and any potential delays due to weekends or holidays is crucial for maintaining customer satisfaction. Providing estimated delivery dates based on real-time data and carrier information helps manage expectations effectively.

G. Considered Discounts and Negotiated Rates

Businesses with high shipping volumes may receive discounted rates, which should be applied to the standard rates.

H. Automate the Process

Shipping software or platforms integrated with carrier APIs can automate rate calculations and provide instant quotes to customers.

Gather data on the business's shipping volume, typically measured as the total number of shipments or total weight shipped over a specified period (e.g., monthly).

Define Discount Tiers:

Establish discount tiers based on shipping volume. For example:

Tier 1: 1-500 shipments/month = 5% discount

Tier 2: 501-1000 shipments/month = 10% discount

Tier 3: Over 1000 shipments/month = 15% discount

Determine Eligibility:

Assess if the business qualifies for a discount based on its shipping volume. Compare the business's volume against the predefined tiers.

Obtain the standard shipping rate from the carrier based on the shipment details, such as package size, weight, and destination. Calculate the discount amount based on the applicable tier.

Discount Amount = Standard Rate * Discount Percentage

Adjust the standard rate by subtracting the discount amount.

Discounted Rate = Standard Rate - Discount Amount

Calculate and add any additional fees, such as fuel surcharges, handling fees, customs duties, and insurance, to the discounted rate. Update the shipping cost information with the discounted rate and additional fees for accurate billing and invoicing. Provide the updated shipping rates to the business, including details about the discount applied and any additional fees.

Continuously monitor the business's shipping volume and update the discount tiers or rates as necessary.

Review and adjust the discount structure periodically based on changes in shipping volume or carrier agreements.

I. Provide Transparency to Customers

Displaying shipping options, costs, and estimated delivery times at checkout improves customer experience. As part of the online delivery service, it's required to provide the customer with proper information on product delivery, builds trust on the business and it improves the repeated customers over the time.

Case Study: Walgreens Online Order Fulfillment

Walgreens, a major retail pharmacy chain, has implemented a robust system for handling online orders and shipping estimation.

A. Online Order Process

1. Order Placement: Customers place orders through the Walgreens website or mobile app.

2. Order Confirmation: An automated system confirms the order and provides an estimated delivery time.

3. Order Processing: The order is processed at the nearest Walgreens distribution center.

B. Shipping Estimation at Walgreens

Walgreens uses a combination of in-house technology and third-party carrier APIs to generate shipping estimates.

C. Use of Carrier APIs

Walgreens integrates with multiple carrier APIs (e.g., FedEx, UPS) to obtain real-time shipping rates and delivery times.

For example, fi you are working with Carrier like FedEx, As a corporate user Walgreens supposed to authenticate with FedEx API and create the authorization certificate Key and that key value supposed to share across the network and create the request with the FedEx with all the details like From Address, To Address, type of shipment and additional details.

This kind of format will be work if the type of order online via website or mobile app. This approach is also completely different if offline orders. As Walgreens also Pharmacy store and it may receive the orders from doctors with the patient's information and this information will be retrieved through the pharmacy portal or systems. In this case we automate the process with the schedule of everyday night after business hours. As part of this automation, we retrieve all the details from respective tables and update the shipping estimation details in respective tables.

Then the actual customers will receive the shipment notification via email or phone message.

J. Inventory Management

Efficient inventory management ensures that products are available for timely shipping. Walgreens uses automated systems to track stock levels and predict demand.

The Era of Automation in Shipping

The era of automation in shipping marks a transformative phase in the logistics industry, characterized by the integration of advanced technologies such as artificial intelligence (AI), machine learning, and robotics. Automation has revolutionized the way shipping processes are managed, from real-time rate calculations and

dynamic route optimization to automated packaging and sorting systems. These innovations enhance efficiency, reduce human error, and significantly cut operational costs. Furthermore, automated tracking and inventory systems provide customers with real-time updates, fostering transparency and improving overall customer satisfaction. As automation continues to evolve, the shipping industry is poised for even greater advancements, paving the way for a more streamlined, cost-effective, and reliable supply chain ecosystem.

Data Transmission Between Warehouse Systems and Shipping Carriers

Data transmission between warehouses and shipping carriers is vital for efficient logistics and supply chain management. Using advanced APIs and secure data exchange protocols, warehouses can seamlessly transmit crucial shipment information such as inventory levels, package details, and destination addresses directly to shipping carriers in real-time. This automated communication ensures accurate rate calculations, efficient routing, and prompt scheduling of pick-ups and deliveries. Real-time data transmission also enables warehouses to provide up-to-date tracking information, enhancing transparency and customer satisfaction. By streamlining these processes, warehouses can significantly reduce errors, optimize operations, and ensure timely and reliable delivery of goods.

Conclusion

Accurate shipping estimation and efficient order fulfillment are essential components of successful e-commerce operations. Walgreens serves as a model for other retailers looking to enhance their online order processes. Future research could explore the impact of emerging technologies such as AI and blockchain on shipping estimation and order fulfillment.

References

[1]. FedEx Corporation, "FedEx API Documentation," [Online]. Available: https://www.fedex.com/en-us/developer.html

[2]. United Parcel Service, "UPS Developer Kit APIs," [Online]. Available: https://www.ups.com/upsdeveloperkit

[3]. Walgreens, "Walgreens Online Order Fulfillment," [Online]. Available: https://www.walgreens.com/topic/help/order.jsp

[4]. “Logistics and Supply Chain Management” by Martin Christopher: A comprehensive book on logistics and supply chain management, including shipping cost estimation.

[5]. “Introduction to Logistics Systems Management” by Gianpaolo Ghiani, Gilbert Laporte, and Randy G. Goetschalckx: Covers logistics, including shipping estimation and cost management.

[6]. “2024 Shipping and Fulfillment Report” by the eCommerce Foundation: Offers insights into trends, challenges, and solutions in shipping and fulfillment.