Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет комп'ютерних систем та мереж
(повне найменування інституту, назва факультету (відділення))
Кафедра обчислювальної техніки
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему Програмний засіб для автономного формування замовлень філіалу торгової платформи Amazon

Виконав: студент 2 курсу, групи КІ-18 м
спеціальності
123– Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)

Матейщук А. А.
(прізвище та ініціали)
Керівник к.т.н. доц. Черняк О. І.
(прізвище та ініціали)
Рецензент к.т.н. доц. Карпінець В. В.
(прізвище та ініціали)

Вінниця ВНТУ – 2019 року

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Інститут Інформаційних технологій та комп'ютерної інженерії
Кафедра Обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Спеціальність _____

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ, д.т.н., проф.
_____Мартинюк Т.Б.
"____" _____20___року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Матейщуку Андрію Анатолійовичу

1) Тема роботи: Програмний засіб для автономного формування замовлень філіалу торгової платформи Amazon.
Керівник роботи: Черняк Олександр Іванович, к.т.н., доцент,

затверджені наказом ВНТУ від _____ 20__ року №__

2) Строк подання студентом роботи_____ 20__ р.
3) Вихідні дані до проекту: розробити програмний засіб для автономного формування замовлень філіалу торгової платформи Amazon.
4) Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. 1. Існуючі інтернет-ресурси. Їх переваги та недоліки. 2. Дослідження розробки програмного засобу. 3. Розробка програмного засобу для обробки і формування запитів на поставку товарів. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5) Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
6) Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Черняк О. І., доц., к.т.н. каф. ОТ		
4	Глущенко Л. Д., доц., к.е.н. каф. ЕПОВ		

7) Дата видачі завдання _____ 20__

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Аналіз завдання. Вступ	15.09.19-18.09.19	
2	Розробка техніко-економічного обґрунтування	19.09.19-26.09.19	
3	Розробка технічного завдання	27.09.19-02.10.19	
4	Аналіз літературних джерел за напрямком роботи	03.10.19-06.10.19	
5	Розробка економічної частини	07.10.19-14.10.19	
6	Аналіз виконання поставленого завдання, висновки	15.10.19-02.11.19	
7	Оформлення пояснювальної записки	03.11.19-12.11.19	
8	Попередній захист роботи	15.11.19	
9	Представлення роботи до захисту	10.12.19	
10	Захист роботи	12.12.19	

Студент _____Матейщук А. А._____

Керівник роботи _____Черняк О. І._____

АНОТАЦІЯ

У даній роботі розглянуто процес автономного створення посилок продавцями перед відправкою їх до покупця з використанням Amazon MWS API. Проведено аналіз існуючих програмних засобів та визначено їх переваги та недоліки. Розроблено метод автономного створення посилок перед їх обробкою.

Покращено процес сканування товарів використовуючи штрих-коди. Розроблено інтерфейс взаємодії користувача з програмою, а також розраховано економічну доцільність створення програмного засобу.

ABSTRACT

This paper examines the process of offline parcel creation by sellers before sending them to the buyer using the Amazon MWS API. The analysis of the existing software tools and their advantages and disadvantages are determined. The method of autonomous creation of parcels before their processing is developed.

Product scanning process has been improved using barcodes. The interface of the user interaction with the program is developed, and also the economic expediency of creation of the software is calculated.

ЗМІСТ

ВСТУП

В еру інформаційних технологій велику популярність набуває ведення бізнесу за допомогою програмного засобу, який дає точну картину товарообігу та статистику купівлі-продажу товарів. Тим самим дозволяє продавцям відслідковувати успішність товарів, потребу повторної закупівлі товару для продовження його постачання покупцям.

Враховуючи потужності комп'ютерних процесорів та їх швидкість легко здогадатись, речі, які пов'язані з веденям бізнесу краще переносити на IT сторону, тим самим давати комп'ютерам виконувати операції, які будуть пораховані за лічені секунди.

Завдяки цьому людина економить час на обрахунки, коли об'єми великі, а також це зменшує ризик виникнення помилок в обрахунках, які користувач може допустити через фактори, які можуть впливати на нього.

На сьогоднішній день існує немала кількість програмних засобів, для обробки запитів на поставку продукції у відділи логістики Amazon. Важливість такого функціоналу зобумовлена тим, щоб Amazon завжди мав товар в наявності, щоб продажі не зупинялись через його відсутність. Тим самим отримуючи та обробляючи запити, забезпечується постійний рух бізнес-процесів, а саме продажів без простоювання товару на складі. Кожне рішення має свої за і проти.

Однак визначити найкраще серед існуючих рішень неможливо, тому як кожне має свої переваги перед своїми конкурентами, а також може мати певний функціонал, що не має конкурент. Бувають ситуації, коли користувачу потрібний функціонал декількох продуктів, що є неможливим доки в одному з програмних рішень він не буде реалізованим.

Ola.com дає достатньо багатий функціонал, невисоку ціну, але не має онлайн-підтримки та форуму, користувачам доводиться писати в меседж-підтримку і в середньому відповіді 3 дні.

Програмне рішення від eBay - eBay AWS, розроблялось 4 програмістами. Має високу ціну відносно своїх конкурентів та не такий багатий функціонал,

однак має шифрування передачі даних між фронт-енд частиною та бек-енд, тим самим виключає можливість хакерам вкрасти інформацію про товари та їх рівень продажів. Підтримки не має, однак є форум, на якому відслідковуються теми, таким чином і виправляються помилки.

uBid sellercloud – програмне забезпечення від uBid.com, має онлайн-підтримку, середню ціну на використання. Також широкий функціонал відносно своїх конкурентів, але користувацький інтерфейс побудований не найкращим чином.

Bonanza sellersnap достатньо нове рішення (з 2016 року), використовує особливий підхід, а саме NoSQL базу даних – mongodb, нехтуючи структурованістю, але виграючи швидкості роботи самого продукту. Очікується друга версія продукту, однак перша показує гарні результати. Достатня невелика ціна, розробники постійно спілкуються з користувачами, беручи від них різні відгуки та побажання до покращення у другій версії.

Amazon sellercentral - скретчпад від Amazon, який є інструментом для використання розробниками Amazon MWS API. Скретчпад допомагає розробникам розібратись як працює API на прикладах, для створення програмних продуктів або для своїх потреб, або для програмної комерції. Дане рішення не є найкращим для людей, які є віддаленими від програмування. Але якщо є розуміння в тому як працює API для простих користувачів, то вони вибирають саме його, так як тут неможливі помилки, які можуть траплятись в програмних рішеннях, які є проміжними між користувачем та Amazon, швидкість обробки є моментальною, так як не доводиться переводити з одного формату в інший для того, щоб відправити запит в Amazon, потім те саме робити, для того, щоб відобразити користувачу відповідь від Amazon.

Даний програмний засіб є простим у використані під час автономного формування посилок за рахунок спрощеного методу створення, не потребує особливих навичок та затрат часу для того, щоб розібратись у роботі з ним.

Існуючі рішення мають широкі функціональні можливості і є високоефективними, але вони є складними для користування та потребують

високого втручання у процес створення. При цьому вони призначені для користувачів, які не мають досвіду користування з подібними продуктами. У цьому полягають їх недоліки. Тому розробка програмного засобу, яке спростить процес створення посилки є **актуальною задачею**.

Об'єктом дослідження є формування замовлень на логістику в Amazon.

Предметом дослідження є програмний засіб для обробки і автономного формування запитів на поставку товарів.

Метою роботи є розбробка програмного засобу формування запитів на логістику в Amazon з розробленим методом створення автономних посилок.

Для досягнення поставленої мети потрібно вирішити такі задачі:

- проаналізувати існуючі програмні засоби, їх переваги та недоліки;
- розробити метод автономного створення посилок;
- покращити сканування товарів використвоючи штрих-коди;
- автоматизувати створення посилок перед їх обробкою;
- розробити інтерфейс взаємодії користувача з програмою;
- розрахувати економічну доцільність створення програмного засобу.

Наукова новизна знайшла подальшого розвитку технології формування замовлення використовуються штрих коди, що дозволило автоматизувати цей процес і скоротити кількість дій оператора.

Практичне значення одержаних результатів полягає в розробці програмногозасобу, який передбачається розмістити на open-source сайтах для безкоштовного використання.

Публікації [7]: Матейщук А. А. Програмний засіб для автономного формування замовлень філіалу торгової платформи Amazon.Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2019)».Збірник матеріалів.- Вінниця, ВНТУ, 2019. -с. 15-16. - [Електронний ресурс]. Режим доступу https://conferences.vntu.edu.ua/public/files/mn/mn-2019_netpub.pdf Дата звернення: листопад, 2019.

1 ІСНУЮЧІ ІНТЕРНЕТ-РЕСУРСИ ДЛЯ КУПІВЛІ-ПРОДАЖУ ТОВАРІВ

Електронна комерція - це діяльність з купівлі або продажу продуктів в онлайн-сервісах. Електронна комерція використовує такі технології, як мобільна комерція, електронний переказ коштів, управління ланцюгами поставок, інтернет-маркетинг, онлайн-обробка транзакцій, електронний обмін даними (EDI), системи управління запасами та автоматизовані системи збору даних.

Електронна комерція зазвичай використовує World Wide Web, принаймні для однієї частини життєвого циклу транзакції, хоча вона також може використовувати інші технології, такі як електронна пошта. Типові транзакції електронної торгівлі включають покупку онлайн-книг і покупку, в меншій мірі, індивідуальні / персоналізовані онлайн-магазини інвентарю. Існує три галузі електронної комерції: онлайн-магазини, електронні ринки і онлайн-аукціони. Електронна комерція підтримується електронним бізнесом.

Підприємства електронної комерції бувають наступних видів:

– забезпечення або участь у онлайн-ринках, які обробляють сторонніх комерційних споживачів або виконують продаж типу від споживача до споживача;

– інтернет-магазини для роздрібних продажів безпосередньо споживачам через веб-сайти та мобільні додатки, а також розмовна торгівля через чат, чат-ботів та голосових помічників;

– купівля-продаж бізнесу;

– збір та використання демографічних даних за допомогою веб-контактів та соціальних мереж;

– обмін електронними даними між підприємствами;

– маркетинг потенційним і встановленим клієнтам електронною поштою або факсом;

– онлайн-фінансові обміни для валютних бірж або торгівельних цілей.

Деякими загальними додатками, пов'язаними з електронною торгівлею, є:

– розмовна торгівля: електронна комерція через чат;

- цифровий гаманець;
- автоматизація документів у ланцюзі поставок та логістиці;
- електронні квитки;
- управління змістом підприємства;
- групова покупка;
- миттєві повідомлення;
- групи новин;
- інтернет-банкінг;
- інтернет-офісні апартаменти;
- інтернет-магазини та відстеження замовлень;
- друк на замовлення;
- кошик програмного забезпечення;
- соціальна мережа;
- телеконференція;
- віртуальний помічник (штучний інтелект)
- вітчизняні та міжнародні платіжні системи.

Ринки електронної комерції ростуть із помітними темпами. Очікується, що інтернет-ринок зросте на 56% у 2015-2020 роках. Традиційні ринки очікують лише 2% зростання того ж часу. Цехи та роздрібні торговці борються через інтернет-магазини пропонувати більш низькі ціни. Багато великих роздрібних торговців можуть підтримувати свою присутність у режимі офлайн і в інтернеті, пов'язавши фізичні та онлайнові пропозиції.

Електронна комерція дозволяє клієнтам подолати географічні бар'єри та дозволяє їм купувати продукти в будь-який час і з будь-якого місця. Інтернет та традиційні ринки мають різні стратегії ведення бізнесу. Традиційні роздрібні торговці пропонують меншу кількість асортиментів продукції через простір, де інтернет-магазини часто не проводять інвентаризацію, а безпосередньо направляють замовлення замовника на виробництво. Стратегії ціноутворення також відрізняються для традиційних та інтернет-магазинів. Традиційні роздрібні

торговці встановлюють свої ціни на трафік магазину та витрати на ведення інвентаризації. Інтернет-магазини базують ціни на швидкості доставки.

Тим не менш, електронна комерція не має взаємодії з людьми для клієнтів, особливо тих, хто віддає перевагу особистому зв'язку. Клієнти також стурбовані безпекою онлайн-транзакцій і, як правило, залишаються лояльними до відомих роздрібних торговців. В останні роки роздрібні торговці одягом, такі як Томмі Хілфігер, почали додавати платформи Virtual Fit на свої сайти електронної комерції, щоб зменшити ризик того, що клієнти купують одяг неправильного розміру, хоча ці варіанти значно відрізняються за своїм призначенням. Коли замовник шкодує про покупку продукту, це передбачає повернення товару та відшкодування. Цей процес незручно, тому що клієнтам потрібно упакувати та відправити товар. Якщо продукти дорогі, великі або тендітні, це стосується питань безпеки.

Електронна комерція зросла у важливості, оскільки компанії прийняли системи просунення та рекламування. Ми можемо відрізняти систему каналів чистого клацання (pure-click), клацання та клацання (brick-and-click), прийняті компаніями:

– компанії, які займаються pure-click є тими, хто запустив веб-сайт без жодного попереднього існування як фірми;

– компанії з brick-and-click – ті існуючі компанії, які додали інтернет-сайт для електронної комерції;

– інтернет-магазини з brick-and-click – ті, які згодом відкрили фізичні адреси, щоб доповнити свої можливості в інтернеті.

Сучасна тенденція електронної комерції рекомендує компаніям змінити традиційну бізнес-модель, в якій основна увага зосереджена на "стандартизованих продуктах, однорідному ринку та тривалому життєвому циклі продукту", до нової бізнес-моделі, де основна увага приділяється "різноманітним та індивідуальним продуктам". Також вимагає, щоб компанія мала здатність задовольнити численні потреби різних клієнтів і надавати їм широкий асортимент товарів.

З більш широким вибором продуктів, інформація про продукти для клієнтів, щоб вибрати і задовольнити їх потреби стають ключовими. Для того, щоб вирішити принцип масової адаптації до компанії, пропонується використовувати систему рекомендацій. Ця система допомагає рекомендувати відповідні продукти для клієнтів та допомагає клієнтам приймати рішення під час процесу покупки. Система рекомендувачів може працювати через провідних продавців на веб-сайті, демографічні показники споживачів або поведінку покупців. Проте існують 3 основних способи рекомендацій: рекомендації продукту безпосередньо клієнтам, надання детальної інформації про продукти та виявлення думок або критичних відгуків інших покупців. Це користь для споживчого досвіду без фізичного шопінгу. Загалом, система рекомендацій використовується для зв'язку з клієнтами в Інтернеті та допомагає знаходити потрібні продукти, які вони хочуть ефективно та безпосередньо [1].

Інтернет-магазини – це форма електронної комерції, яка дозволяє споживачам безпосередньо купувати товари або послуги від продавця через Інтернет за допомогою веб-браузера. Споживачі знаходять продукт, що представляє інтерес, відвідавши веб-сайт продавця безпосередньо або шукаючи серед альтернативних постачальників, використовуючи пошукову систему пошуку покупок, яка відображає наявність та вартість того самого продукту у різних електронних роздрібних торговців.

Інтернет-магазин викликає фізичну аналогію покупки товарів або послуг у звичайному роздрібному торговельному або торговельному центрі "brick-and-mortar"; цей процес називається інтернет-покупок "бізнес-споживач" (B2C). Коли інтернет-магазин встановлюється, щоб дозволити підприємствам купувати покупки в іншому бізнесі, цей процес називається бізнес-бізнес (B2B) для онлайн-покупок. Типовий інтернет-магазин дає змогу клієнту переглядати спектр продуктів та послуг фірми, переглядати фотографії або зображення продуктів, а також інформацію про технічні характеристики, характеристики та ціни продукції.

Інтернет-магазини зазвичай дозволяють покупцям використовувати функції пошуку для пошуку конкретних моделей, брендів або предметів. Інтернет-клієнти повинні мати доступ до Інтернету та дійсний спосіб оплати, щоб завершити транзакцію, таку як кредитна картка, дебетова картка з підтримкою Interac або така послуга, як PayPal. Для фізичних продуктів (наприклад, книжок з м'якушками або одягу) e-tailer відправляє продукти клієнту; Для цифрових продуктів, таких як цифрові аудіофайли пісень або програмного забезпечення, e-tailer зазвичай надсилає файл клієнту через Інтернет. Найбільшими з цих корпоративних роздрібних мереж є Alibaba, Amazon.com та eBay.

Альтернативними назвами діяльності "e-tailer" є скорочена форма " електронного шопінгу" чи скорочена форма "електронних магазинів". Інтернет-магазин можна назвати електронним інтернет-магазином, електронним магазином, інтернет-магазином, веб-магазином або віртуальним магазином. Мобільна комерція (або m-commerce) описує придбання веб-сайту або програмного забезпечення, оптимізованого для мобільних пристроїв в мережі інтернет. Ці веб-сайти або додатки призначені для того, щоб користувачі могли переглядати товари та послуги компаній на планшетних комп'ютерах та смартфонах.

Згодом ризик і довіра також стануть двома важливими чинниками, що впливають на поведінку людей у цифрових середовищах. Клієнт розглядає можливість перемикання між електронними каналами, оскільки вони в основному впливають на порівняння з автономними магазинами, що включає зростання рівня безпеки, фінансових та ефективних ризиків. Іншими словами, покупці в Інтернеті через Інтернет можуть отримати більше ризику, ніж покупці в магазинах. Існує три фактори, які можуть вплинути на людей для прийняття рішення про покупку, по-перше, люди не можуть перевірити, чи відповідає продукт своїм потребам і потребам, перш ніж вони отримують це рішення. По-друге, клієнт може турбуватися про послуги післяпродажного обслуговування. Нарешті, клієнт може побоюватися, що вони не зможуть повністю зрозуміти мову, яка

використовується в електронних продажах. Виходячи з цих факторів, клієнт сприймає ризик може значною мірою вплинути на поведінку онлайн-покупки.

Інтернет-магазини зосереджують увагу на аспекті довіри до клієнтів, і довіра - це ще один спосіб керування поведінкою клієнтів у цифровому середовищі, що залежить від ставлення та очікувань споживачів. Дійсно, дизайн продукції компанії або ідеї не можуть задовольнити очікування клієнтів. Спонукання покупця на основі раціональних очікувань, а також впливає на емоційну довіру. Крім того, ці очікування також можуть бути встановлені на інформаційній базі продукту та перегляді від інших.

Споживачі знаходять продукт, який цікавить їх, відвідавши веб-сайт продавця безпосередньо або шукаючи серед альтернативних постачальників, використовуючи пошукову систему покупки. Коли певний продукт знайдено на веб-сайті продавця, більшість інтернет-магазинів використовують програмне забезпечення для кошика, щоб споживач міг накопичити кілька предметів та скорегувати кількість. Процес "перевірки" в якому збирається інформація про оплату та доставку.

Деякі магазини дозволяють споживачам зареєструватися для постійного онлайн-рахунку, так що частину або всю цю інформацію потрібно ввести лише один раз. Споживач часто отримує підтвердження електронною поштою після завершення транзакції. Менш складні магазини можуть покладатися на споживачів, щоб телефонувати або надіслати свої замовлення електронною поштою (хоча цілі номери кредитних карток, термін придатності та код безпеки, або банківський рахунок та номер маршрутизації не повинні передаватись електронною поштою з міркувань безпеки).

Інтернет-покупці часто використовують кредитну картку або обліковий запис PayPal для здійснення платежів. Однак деякі системи дозволяють користувачам створювати облікові записи та оплачувати альтернативні способи, такі як:

– платіж за мобільними телефонами та стаціонарними лініями;

– готівка на доставці;

– дебетова картка;

– прямий дебет в деяких країнах;

– електронні гроші різних типів;

– подарункові карти;

– поштовий грошовий переказ;

– передача / доставка за платіж;

– рахунок-фактура, особливо популярний на деяких ринках / країнах, таких як Швейцарія;

– біткойн або інша криптова валюта.

Деякі інтернет-магазини не приймуть міжнародні кредитні картки. Деякі вимагають, щоб як платіж покупця, так і адреса доставки знаходилися в тій самій країні, що і робота бази інтернет-магазину. Інші інтернет-магазини дозволяють клієнтам з будь-якої країни надсилати подарунки будь-де.

Фінансова частина транзакції може оброблятися в режимі реального часу (наприклад, дозволяючи споживачеві знати, що їхня кредитна картка була відхилена до відключення) або може бути виконана пізніше як частина процесу виконання.

Як тільки платіж буде прийнятий, товари або послуги можуть бути доставлені наступним чином.

Для фізичних предметів:

– доставка - продукт відправляється на вказану клієнтом адресу. Доставка роздрібних пакетів, як правило, здійснюється державною поштовою системою або роздрібним кур'єром;

– передача відгруз - замовлення передається виробнику або дистриб'ютору сторонніх виробників, який потім передає товар безпосередньо споживачеві, обходячи фізичне розташування продавця, щоб заощадити час, гроші та простір;

– забирання в магазині - клієнт вибирає місцевий магазин, використовуючи програмне забезпечення локатора та збирає доставлений продукт у вибраному місці. Цей спосіб часто використовується в бізнес-моделі brick-and-click.

Для цифрових товарів чи квитків:

– завантаження - Метод часто використовується для цифрових мультимедійних продуктів, таких як програмне забезпечення, музика, фільми або зображення.

– друкування, надання коду або електронної розсилки таких предметів, як вхідні квитки та квитки (наприклад, подарункові сертифікати та купони). Квитки, коди або купони можуть бути викуплені у відповідних фізичних чи онлайн-приміщеннях, а їх зміст переглядається для підтвердження їх відповідності (наприклад, запевненням, що право на вхід або використання викуплено у правильний час та місце для правильної суми долара , і для правильної кількості використання).

– телефонне замовлення або "пікап біля дверей" - покупець забирає попередньо придбані квитки на подію, таку як п'єса, спортивна подія або концерт, безпосередньо перед подією або заздалегідь З настанням сайтів Інтернет та електронної комерції, які дозволяють клієнтам купувати квитки в Інтернеті, популярність цієї послуги збільшилася.

Замовники приваблюють покупки в Інтернеті не тільки через високий рівень зручності, але і за рахунок більш широкого вибору, конкурентного ціноутворення та більшого доступу до інформації. Бізнес-організації намагаються запропонувати інтернет-магазини не лише тому, що вони значно дешевші, ніж звичайних магазинів, але і тому, що вони забезпечують доступ до світового ринку, підвищують вартість клієнта та створюють стійкі можливості.

Дизайнери інтернет-магазинів стурбовані наслідками навантаження інформації. Інформаційне навантаження є продуктом просторового та часового розташування подразників у веб-магазині. Порівняно з традиційними

роздрібними покупками, інформаційне середовище віртуального шопінгу посилюється, надаючи додаткову інформацію про продукт, таку як порівняльні продукти та послуги, а також різні альтернативи та атрибути кожної альтернативи тощо. Два основних виміри інформаційного навантаження - це складність та новизна. Складність - це кількість різних елементів або особливостей сайту, часто результат збільшення різноманітності інформації. Новизна передбачає несподівані, пригнічені, нові або незнайомі аспекти сайту. Вимірювання новизни може дозволити споживачам вивчати торговий сайт, тоді як складність може викликати імпульсивні покупки.

Згідно з результатами дослідницької доповіді Університету Western Michigan, опублікованого в 2005 році, веб-сайт електронної комерції не повинен виглядати добре з переліком на багатьох пошукових системах. Він повинен будувати відносини з клієнтами, щоб заробляти гроші. У звіті також йдеться про те, що веб-сайт повинен залишити позитивне враження для клієнтів, що дає їм привід повернутися. Проте дослідження довели, що сайти з більшою увагою до ефективності, зручності та персоналізованих послуг збільшили мотивацію покупців робити покупки.

Компанія Dyn провела опитування понад 1400 споживачів в 11 країнах у Північній Америці, Європі, на Близькому Сході та в Азії, а результати опитування були такими:

– інтернет-магазини повинні покращити швидкість своїх веб-сайтів;

– інтернет-магазини повинні полегшити споживачам страх, який обумовлений через не впевненість отримання замовленого товару.

Ці проблеми найбільше впливають на рішення майже двох третин споживачів.

Найвідомішими серед них є наступні:

– eBay;

– Amazon;

– uBid;

– Bonanza;

– Onlineauction (OLA).

Розглянемо переваги і недоліки кожного маркету.

Amazon і eBay являються гігантами по оборотоу товару та прибутку серед усіх інтернет-маркетів, де покупці зі всього світу роблять замовлення кожного дня, а продавці використовують ці ресурси для розвитку свого товару, збільшення кількості клієнтів, і, як результат, збільшення продажів. Компанії являються найбільшими конкурентами, хоча способи реалізації товара і методи їх діяльності багато в чому відрізняються.

eBay – інтернет-аукціон, який був створений для продажу товарів між людьми, тобто «з рук в руки». Але зі збільшенням популярності і кількості покупців, ця площадка почала також залучати виробників товарів та інтернет магазини, для яких eBay став якщо не основним, то дуже важливим і прибутковим каналом збуту. eBay можно порівнювати з великим міжнародним ринком, на якому як компанії так і звичайні люди пропонують свій товар усьому світу. Багатонаціональна корпорація електронної комерції, розташована в Сан-Хосе, штат Каліфорнія, яка сприяє продажам з рук в руки через веб-сайт. eBay – багатомільяровий доларовий бізнес з операціями в 30 країнах, починаючи з 2011 року. Компанія керує сайтом eBay.com, онлайновим аукціоном та торговим веб-сайтом, на якому люди та компанії купують та продають різноманітні товари та послуги у всьому світі. Веб-сайт можна вільно використовувати для покупців, але у продавців стягують плату за списки товарів після обмеженої кількості безкоштовних списків і знову, коли ці товари продаються.

Окрім звичайних аукціоних продажів, веб-сайт з тих пір розвивався та розширювався, що включав в себе покупки "Купити зараз"; Покупки через UPC, ISBN чи SKU (через Half.com); Інтернет-класифікована реклама (через Kijiji або eBay Classifieds); торгівля квитками на онлайн-події (через StubHub); та інші послуги. Раніше він пропонував онлайн-грошові перекази (через PayPal, який був дочірньою компанією eBay з 2002 до 2015 року).

eBay генерує прибуток складною системою плати за послуги, що включає в себе характеристики продукту та вартість кінцевої вартості від продажу продавців. Станом на листопад 2012 року американська eBay.com стягує від 0,10 до 2 доларів США, засновану на початковій або резервній ціні, як плату за вхід в основний список аукціонів без будь-яких прикрас. Плата за кінцеву вартість становить 10% від загальної суми продажу, тобто ціна товару плюс витрати на доставку. Запис із фіксованою ціною має плату за вставку в розмірі 0,30 дол., А вартість кінцевої вартості залежить від категорії та загальної суми продажу (наприклад, 13% для DVD та фільмів - до 50 дол. США). Британський eBay.co.uk займає від 0,15 до 0,3 до максимальної ставки £ 3 за £ 100 для звичайної лістингу та до 10% від кінцевої ціни. Знижені кінцеві вартість доступні зареєстрованим клієнтам.

Під час встановлення аукціону eBay надає продавцям можливість вибору способу доставки: звичайною поштою, експрес-поштою та / або кур'єрською службою. Продавець може вибрати для покупців лише один спосіб доставки; або продавець може запропонувати покупцям вибір варіантів. Надзвичайно дешеві предмети, що постачаються безпосередньо з Китаю, іноді доставляються поштовими відправленнями (морською поштою), що є недорогим, але займає від одного до двох місяців. Якщо покупець поспішаєто він може заплатити додатковий внесок, щоб перейти на доставку другого класу "Surface Air Lifted" або на доставку першокласною авіапоштою. Починаючи з 2012 року, eBay залучила продавців до своєї "Global Shipping Program". Якщо продавець використовує програму, стороннім покупцям платять платню в Pitney Bowes. Продавець надсилає товар до об'єкта Pitney Bowes у США (або Великобританії), який потім передає його покупцеві, доглядаючи за усіма міжнародними вимогами доставки. Заявлено, що програма покращує вибір продуктів, доступний для міжнародних покупців.

Загальні критичні зауваження eBay включають в себе політику вимагання використання PayPal для виплат та заклопотаності щодо шахрайства, підробки та порушень прав інтелектуальної власності в аукціонних предметах. Є також

питання про те, як негативні відгуки після аукціону можуть компенсувати переваги використання eBay як торговельну платформу. eBay була піддана критиці за те, що не сплачує податкові декларації у Великобританії: у жовтні 2012 року компанія Sunday Times повідомила, що eBay платив лише 1,2 мільйона фунтів стерлінгів з податку на продаж понад 800 мільйонів фунтів стерлінгів [3].

eBay.com є сьомим за відвідуваністю сайтом в США, багато хто з відвідувачів можуть покупцями, якщо застосовувати правильну тактику продажів і пропонувати потрібний і в той час неконкурентний продукт.

Для продажу рідкісного або унікального товару система аукціону може стати кращим рішенням, а ціна продажу перевищити очікуваний дохід. Якщо товар масово вироблений, його краще продавати за фіксованою ціною.

Система покупок через eBay є досить безпечною, захищаючи обидві сторони торгової угоди.

Більшість операцій купівлі-продажу товару здійснюється через платіжну систему PayPal, яка належить eBay. Хоча це і збільшує безпеку обох сторін, але так само викликає деякі проблеми, такі як утримання ваших коштів, блокування рахунку або неможливість приймати гроші в деяких країнах [4].

Amazon Inc. - американська компанія електронної комерції та хмарних обчислень, яка базується у Сіетлі, штат Вашингтон, заснована Джефом Безосом 5 липня 1994 року. Технічний гіган, що є найбільшим інтернет-магазином у світі, що виміряний за доходами та ринковою капіталізацією, другий за величиною після AlibabaGroup в загальному обсязі продажів.

Компанія Amazon має роздільні веб-сайти для Сполучених Штатів, Великобританії та Ірландії, Франції, Канади, Німеччини, Італії, Іспанії, Нідерландів, Австралії, Бразилії, Японії, Китаю, Індії та Мексики. У 2016 році були також запущені голландські, польські та турецькі мовні версії німецького сайту Amazon. Також пропонує міжнародну доставку деяких своїх продуктів до деяких інших країн.

Лінійки продуктів Amazon.com, доступні на його веб-сайті, містять декілька засобів масової інформації (книги, DVD-диски, музичні компакт-диски,

відеокасети та програмне забезпечення), одяг, дитячі товари, побутові електроніки, косметичні товари, продукти для гурманів, продукти харчування, товари для здоров'я та особистої гігієни, наукові матеріали, кухонні предмети, ювелірні прикраси, годинники, продукти газону та саду, музичні інструменти, спортивні товари, інструменти, автомобільні предмети та іграшки та ігри.

До 2008 року домен amazon.com залучив щонайменше 615 мільйонів відвідувачів. До початку 2016 року Amazon залучить понад 130 мільйонів клієнтів на свій веб-сайт США. Компанія також вклала величезний внесок у величезну кількість серверних потужностей для свого веб-сайту, особливо для обробки надмірного трафіку під час святкових різдвяних свят.

Результати, отримані пошуковою системою Amazon, частково визначаються рекламними комісіями

Амазон в даний час готується до Індії, щоб грати роль у роздрібному секторі продуктових магазинів, спрямованих на забезпечення потреб споживачів.

Amazon.com пропонує ряд продуктів та послуг, зокрема:

- AmazonFresh;
- Amazon Prime;
- Amazon Web Services;
- Alexa;
- Appstore;
- Amazon Drive;
- Echo;
- Kindle;
- Fire tablets;
- Fire TV;
- Video;
- Kindle Store;
- Music;
- Music Unlimited;

– Amazon Digital Game Store;

– Amazon Studios;

– AmazonWireless.

Amazon дозволяє користувачам подавати відгуки на веб-сторінку кожного продукту. Рецензенти повинні оцінювати товар за шкалою від 1 до 5 зірок. Amazon надає можливість відмітити рецензентів, які вказують справжнє ім'я рецензента (на підставі підтвердження облікового запису кредитної картки) або які вказують на те, що рецензент є одним із найкращих рецензентів за популярністю. Клієнти можуть коментувати чи голосувати за відгуки, вказуючи, чи виявили вони корисне для них у огляді. Якщо огляду надається достатня кількість "корисних" звернень, вони з'являються на першій сторінці продукту. У 2010 році Amazon повідомляється як найбільше єдине джерело оглядів інтернет-споживачів.

Щоб уникнути порушень авторських прав, Amazon не повертає машиночитаний текст книги. Замість цього він повертає зображення відповідної сторінки, наказує веб-переглядачу вимкнути друк і обмежує кількість сторінок у книзі, доступ до якої може отримати один користувач. Крім того, клієнти можуть придбати онлайн-доступ до деяких тих самих книг за допомогою програми "Amazon Upgrade".

Amazon отримує багато своїх продажів (близько 40% у 2008 році) від сторонніх продавців, які продають продукти на Amazon. Партнери отримують комісію за пересилання клієнтів до Amazon, розміщуючи посилання на Amazon на своїх веб-сайтах, якщо реферал призводить до продажу. У всьому світі Amazon має "понад 900 000 учасників" у своїх партнерських програмах. У середині 2014 року Партнерська програма Amazon використовується на 1,2% всіх веб-сайтів, і вона є другою за популярністю рекламною мережею після оголошень Google. Він часто використовується веб-сайтами та некомерційними організаціями, щоб забезпечити спонсорів можливість заробляти їм комісію. У 2007 році компанія Amazon повідомила, що більш ніж 1,3 мільйона продавців продали продукти

через сайти Amazon. На відміну від eBay, продавці Amazon не повинні зберігати окремі платіжні рахунки; всі платежі обробляються компанією Amazon.

Amazon.com це, перш за все, роздрібний сайт з моделлю продажів; Amazon приймає незначний відсоток від ціни продажу кожного товару, який продають через його веб-сайт, а також дозволяють компаніям рекламувати свою продукцію, сплачуючи, щоб бути зареєстрованими як продукти.

Політика зі збору податку з продажів Amazon змінилася протягом багатьох років, оскільки на початку діяльності компанія не збирала податок на продаж. У США державні та місцеві податки з продажів стягуються державними та місцевими органами влади, а не на федеральному рівні. У більшості країн, де діє Amazon, податок з продажів або податок на додану вартість є єдиними по всій країні, і Amazon зобов'язаний збирати його від усіх клієнтів. Прихильники змушують Amazon.com збирати податок з продажу - принаймні в штатах, де він підтримує фізичну присутність - стверджують, що корпорація володіє антиконкурентною перевагою перед магазинами, що змушені збирати податок з продажу [5].

У свою чергу, Amazon - це велика торгова площадка, де реалізуються не тільки товари від продавців, але так само і від самого Амазону. Завдяки спільним правилам та умовам, він нагадує великий торговий центр в інтернеті, де представлено безліч магазинів, які продають свої товари і розділені за категоріями. Amazon бере на себе логістику до покупця і складське зберігання товару, а також контролює якісні показники товару, що продається. Якщо з'являється продавець недоброякісного товару, Amazon тут же забирає його зі свого майданчика.

Завдяки системі Amazon, яка не тільки запобігає можливому шахрайству з боку клієнтів, але і гарантує безпеку їх покупок, у більшості людей виникає додатковий рівень довіри і вони розташовані до використання більш надійного Amazon в порівнянні з конкурентами. Компанія гарантує якість товару і його своєчасну доставку, адже всі продавці проходять дуже сувору перевірку і контроль.

В той же час присутні і мінуси Amazon. На Amazon представлена величезна кількість продавців, що значно підвищує рівень конкуренції.

Можна зустріти численні прецеденти, коли Amazon закриває магазин продавця без будь-яких очевидних причин. Всього кілька негативних відгуків незадоволених клієнтів може викликати великі проблеми, навіть якщо вони є маленьким відсотків від усіх покупців. Тому дуже важливо задовольнити запит кожного клієнта, що представляє собою дуже важке завдання [6].

uBid.com - це онлайн-аукціон та веб-сайт із закупівлями на фіксовану ціну, який пропонує як товари, що продаються безпосередньо компанією, так і предмети, що продаються попередньо затвердженими третіми сторонніми торговцями, які сертифікують uBid [7].

Гарна альтернатива Amazon, але через конкуренцію, кількість покупців значно менша, тим самим використання даного ресурсу для ведення бізнесу не підходить великим продавцям [8, 9].

Працює за тим самим принципом, що й eBay. В свою чергу має доволі низькі тарифи на послуги. Але сама компанія недосконало рекламує свій ресурс, тим самим залишає покупців на поглинання гігантами.

Bonanza – це онлайн-ринок, який дозволяє своїм користувачам продавати все, від антикваріату до електроніки. Bonanza подібний до eBay, але його орієнтація на унікальні елементи. Компанія стверджує, що подібні предмети є такими, що можуть бути знайдені на вулиці, будь-який товар доступний на сайті [10].

Bonanza пропонує вести переговори між клієнтом та продавцем в реальному часі, дозволяючи проглядати різноманітні товари віртуально, за допомогою онлайн-трансляції. Крім того, інтернет-аукціон доволі детально описує види оплати, способи доставки і всі інші характеристики та послуги, які потрібно знати покупцю.

Продавці на сайті Bonanza.com можуть заощадити до 50 відсотків на товари, які вони хочуть придбати, за власними продажами. Сайт також пропонує "безкоштовні", в яких продавці погоджуються видавати певні товари з покупками

на певну суму.

Bonanza.com зазвичай описується як альтернатива eBay, і на кінець 2010 року зареєстровано 300 000 користувачів та 3,4 млн. одиниць продажу [11].

OLA.com являє собою інтернет-аукціон, на якому можна продати будь-який товар.

Щоб розпочати торгівлю, необхідно внести певну суму, а також певну плату з кожного продажу. Є найменш відомим ресурсом, серед усіх наведених, тому і кількість нових оголошень за день не є великою [12].

У таблиці 1 наведена порівняльна характеристика вище описаних інтернет-ресурсів.

Таблиця 1.1 – Порівняльна характеристика інтернет-маркетів.

	Amazon	eBay	Ubid	Bonanza	Onlineauction
Логістика	+	-	-	-	-
Зв'язок з клієнтом	-	-	-	+	-
Висока відвідуваність	+	+	-	-	-
Низькі тарифи на послуги	-	+	+	+	+
Високий відсоток конверсії	+	+	-	-	-

У данному розділі було розглянуто існуючі програмні рішення, їх переваги та недоліки. В результаті було визначено недосконаліть існуючих рішень, які потребують більшу кількість дій оператора. Зроблено висновок, що кожен з представлених ресурсів має свої переваги та недоліки, але плюсом Amazon є те, що на нього можна повністю перекласти логістику: достатньо відправити контейнер з товаром і після цього відправляти API запити на відправку конкретного товаруклієнту, а роботи в свою чергу зроблять самі свою справу.

2 ДОСЛІДЖЕННЯ РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ

Товари, які будуть відправлені в посилці до виконавчих центрів Amazon варто вибирати на рівні їх продажів. Тобто, якщо товар продається у великій кількості та прибуток з кожної одиниці товару є значним – його слід включати до посилки і слідкувати, щоб товар був завжди у наявності. Адже тривала відсутність тяне за собою зменшення рівня продажів, тим самим породжуючи втрату прибутку, що є великою помилкою в сфері інтернет-торгівлі.

Партнери можуть отримати доступ до каталогу Amazon безпосередньо на своїх веб-сайтах, використовуючи службу XML Amazon Web Services (AWS). Новий афілійований продукт aStore дозволяє Associates вставляти підмножину продуктів Amazon на інший веб-сайт або зв'язати його з іншим веб-сайтом. У червні 2010 року було розпочато виробництво пропозицій про продавця Amazon (за чутками, що вони отримали внутрішню назву "Project Genesis"), щоб забезпечити більшу прозорість для продавців, рекомендуючи продавати певні продукти стороннім продавцям на Amazon. Рекомендовані продукти засновані на історії перегляду веб-сторінок клієнтів.

Логістика в електронній комерції переважно стосується виконання. Інтернет-ринки та роздрібні торговці мають знайти найкращий спосіб заповнювати замовлення та доставляти продукти. Малі компанії, як правило, контролюють свою власну логістичну операцію, оскільки вони не мають можливості найняти зовнішню компанію. Найчастіше великі компанії наймають службу виконання, яка піклується про логістичні потреби компанії.

2.1 Розробка методу автономного створення посилок.

Розробляючи метод варто звернути увагу на проблему існуючого. Для створення посилки користувачівикористовують метод пакування різних продуктів для однієї коробки. Цей метод має недолік в тому, що користувач не може друкувати наліпки для товарів до того, доки не налаштує всі параметри коробки. Це породжує плутанину протягом робочого процесу, тому як

користувач перериваючи процес може невдало його закінчити після продовження.

Наступною проблемою є те, що друк наклейок відбувається так, що в межах одного листа А4 можуть знаходитись наліпки не лише для одного товару, що створює додаткові часові витрати для того, щоб ці наклейки відсортувати і не перемішати з іншими.

При продажу товарів, які легко підробляються, дуже легко зустріти несподіваних конкурентів, які можуть продавати той же товар за нижчими цінами. Цей факт в деякій мірі впливає на рейтинг аукціону, а тому дуже важливо пропонувати якісний продукт.

Створюючи пакунки користувач має вказавти їх кількість та розміри. Цей процес є занадто заплутаним. Так як потрібно вказувати для кожної коробки, які товари будуть в ній, кількість товарів, а також вагу пакунку. На цьому етапі користувач може зробити забагато помилок.

Прості системи кошиків для покупок (shoppingcart) дозволяють здійснювати офлайнове адміністрування продуктів і категорій. Потім магазин створюється як файли HTML та графіки, які можна завантажити на веб-простір. Системи не використовують онлайнову базу даних. Рішення високого класу можна придбати або взяти на прокат як самостійну програму або як додаток до програми планування ресурсів підприємства. Він зазвичай встановлюється на веб-сервері компанії та може інтегруватися в існуючу ланцюжок поставок, завдяки чому замовлення, оплата, доставка, облік і складування можуть бути в значній мірі автоматизовані.

Інші рішення дозволяють користувачеві реєструватися і створювати інтернет-магазин на порталі, де одночасно розміщується кілька магазинів з одного відділення. Приклади BigCommerce, Shopify та FlickRocket. Пакети кошику з відкритим вихідним кодом включають розширені платформи, такі як Interchange, та інші рішення, такі як Magento, osCommerce, Shopgate, PrestaShop та Zen Cart. Комерційні системи також можуть бути адаптовані, тому магазин не потрібно створювати з нуля.

Використовуючи існуючу структуру, програмні модулі для різних функцій, які вимагає веб-магазин, можуть бути адаптовані та об'єднані.

Маркетинг навколо цифрового середовища, поведінка покупців може не впливати і контролюватися брендом та фірмою, коли вони приймають рішення щодо покупки, яке може стосуватися взаємодії з пошуковою системою, рекомендаціями, оглядами в Інтернеті та іншою інформацією. За допомогою швидкого відокремлення середовища цифрових пристроїв люди частіше використовують свої мобільні телефони, комп'ютери, планшети та інші цифрові пристрої для збору інформації.

Іншими словами, цифрове середовище все більше впливає на думку споживача та купівлю поведінки. В умовах інтернет-магазину інтерактивне рішення може впливати на прийняття рішень щодо надання допомоги клієнтам. Кожен клієнт стає більш інтерактивним, і хоча онлайнові відгуки клієнти можуть впливати на поведінку інших потенційних покупців.

Інтернет-клієнти повинні мати доступ до Інтернету та дійсний спосіб оплати, щоб завершити транзакцію. Як правило, вищі рівні освіти та особистий дохід відповідають більш сприятливим уявленням про покупки в Інтернеті. Збільшення впливу технології також збільшує ймовірність розвитку сприятливого ставлення до нових торгових каналів.

Продаючи за допомогою власного інтернет-магазину, можна непередбачено збільшити маркетингові витрати, що негативно позначиться на прогнозі і розрахунку майбутнього прибутку. З іншого боку, Amazon стягує комісію за кожну одиницю проданого товару, що зазвичай становить фіксований відсоток від ціни. Дуже високий відсоток конверсії - 13%.

Товари, які будуть відправлені в посилці до виконавчих центрів Amazon варто вибирати на рівні їх продажів. Тобто, якщо товар продається у великій кількості та прибуток з кожної одиниці товару є значним – його слід включати до посилки і слідкувати, щоб товар був завжди у наявності. Адже тривала відсутність тяне за собою зменшення рівня продажів, тим самим породжуючи втрату прибутку, що є великою помилкою в сфері інтернет-торгівлі.Також для

кожного пакунку потрібно завантажити перелік товарів. Проблема заключається в тому, що потрібно мати окреме місце на жосткому диску для зберігання тимчасових PDF-файлів. Після цього відкривати кожен файл, далі брати коробку, наступним кроком знаходити потрібні товари, віризати наліпки та клеїти їх на товари та закривати коробку. Даний процес має велику залежність від користувача.

Таким чином кількість дій, які потрібно виконати оператору у стандартному процесі формування замовлень для відправлення в середньому становить 57-63 [15]. Хоча це можливо спростити і виконувати усі обрахунки на комп'ютерному рівні.

2.2 Покращення сканування товарів використвоючи штрих-коди

Штрих-коди прискорюють процес пошуку товару. Наприклад, користувачу потрібно здійснити пошук товару. Замість того, щоб вводити з клавіатури назву товару, достатньо просканувати раніше підготовлений штрих-код. Можна дописати логіку дешифрування таким чином, щоб декодувати також кількість предметів, тощо.

Коли діло підходить до придбання в інтернеті різноманітних товарів, більшість людей першим ділом шукають товар через пошукові системи і після цього потрапляють на різні інтернет-маркети.

Існує два способи для продавців вести бізнес через електронну комерцію: повністю онлайн або онлайн разом з офлайн магазином. Інтернет-продавці можуть запропонувати більш низькі ціни, більший вибір продуктів і високі показники ефективності. Багато клієнтів вибирають інтернет-ринки, якщо продукти можуть бути швидко доставлені за відносно низькою ціною. Проте інтернет-магазини не можуть запропонувати фізичний досвід, який можуть отримати традиційні роздрібні торговці. Важко судити про якість продукту без фізичного досвіду, що може спричинити непередбачуваність споживачів або продавців. Ще одне питання щодо онлайн-ринку - це занепокоєння щодо безпеки онлайн-транзакцій. Через цю проблему багато клієнтів залишаються лояльними до відомих роздрібних торговців.

Безпека є основною проблемою електронної комерції у розвинутих та країнах, що розвиваються. Безпека електронної комерції захищає веб-сайти компаній та споживачів від несанкціонованого доступу, використання, зміни або знищення. Тип загроз включає: шкідливі коди, небажані програми (рекламні продукти, шпигунські програми), фішинг, хакерство та кібер-вандалізм. Веб-сайти електронної комерції використовують різні інструменти для запобігання загрозам безпеці. Ці інструменти включають брандмауери, програмне забезпечення для шифрування, цифрові сертифікати та паролі.

Тривалий час компанії були занепокоєніперевагою технологією ланцюга постачання та рішеннями щодо цієї переваги. Проте поява електронної комерції забезпечила більш практичний та ефективний спосіб надання переваг новим технологіям постачання.

Електронна комерція має можливість інтегрувати всі міжфункціональні та внутрішньофірмові функції, а значить, електронна комерція може вплинути на три потоки (фізичний потік, фінансовий потік та інформаційний потік) ланцюга постачання. Захоплення фізичними потоками покращило рівень руху продукції та інвентарю для компаній. Для інформаційних потоків електронна комерція оптимізувала можливості обробки інформації, ніж компанії, що використовуються, а для фінансових потоків електронна комерція дозволяє компаніям мати більш ефективні рішення щодо оплати та розрахунків.

Крім того, електронна комерція має більш витончений рівень впливу на ланцюжки постачань: по-перше, розрив у продуктивності буде виключено, оскільки компанії можуть визначати прогалини між різними рівнями ланцюгів постачання за допомогою електронних рішень; по-друге, внаслідок появи електронної комерції нові можливості, такі як впровадження систем ERP, таких як SAP ERP, Xero або Megaventory, допомогли компаніям керувати операціями з клієнтами та постачальниками. Однак ці нові можливості поки що не використовуються повністю; по-третє, технологічні компанії продовжують інвестувати в нові програмні рішення для електронної комерції, оскільки очікують повернення інвестицій; по-четверте, електронна комерція допоможе

вирішити багато аспектів проблем, з якими компанії можуть зіткнутися, наприклад, політичні бар'єри або зміни між країнами. Нарешті, електронна комерція надає компаніям більш ефективний спосіб співпрацювати один з одним в межах ланцюга поставок.

Електронна комерція допомагає створювати нові можливості для роботи через інформаційні послуги, програмне забезпечення та цифрові продукти. Це також призводить до втрати робочих місць. Найбільш передбачуваними втратами робочих місць є торгові, поштові та туристичні компанії. Розвиток електронної комерції створить робочі місця, які потребують висококваліфікованих працівників для управління великою кількістю інформації, вимогами клієнтів та виробничими процесами. Навпаки, люди з низькими технічними навичками не можуть користуватися добробутом заробітної плати. З іншого боку, оскільки електронна комерція вимагає достатніх запасів, які можуть бути доставлені клієнтам у часі, склад стає важливим елементом. Склад повинен мати більше співробітників, щоб керувати, контролювати та організовувати, таким чином, умови складу середовища будуть стурбовані співробітниками.

В еру інформаційних технологій велику популярність набуває ведення бізнесу за допомогою програмного засобу, який дає точну картину товарообігу та статистику купівлі-продажу товарів. Тим самим дозволяє продавцям відслідковувати успішність товарів, потребу повторної закупівлі товару для продовження його постачання покупцям.

Враховуючи потужності комп’ютерних процесорів та їх швидкість легко здогадатись, речі, які пов’язані з веденям бізнесу краще переносити на ІТ сторону, тим самим давати комп’ютерам виконувати операції, які будуть пораховані за лічені секунди.

Завдяки цьому людина економить час на обрахунки, коли об’єми великі, а також це зменшує ризик виникнення помилок в обрахунках, які користувач може допустити через фактори, які можуть впливати на нього.

На сьогоднішній день існує немала кількість програмних засобів, для обробки запитів на поставку продукції у відділи логістики Amazon. Важливість

такого функціоналу зобумовлена тим, щоб Amazon завжди мав товар в наявності, щоб продажі не зупинялись через його відсутність. Тим самим отримуючи та обробляючи запити, забезпечується постійний рух бізнес-процесів, а саме продажів без простоювання товару на складі. Кожне рішення має свої за і проти.

З більш широким вибором продуктів, інформація про продукти для клієнтів, щоб вибрати і задовольнити їхні потреби стають ключовими. Для того, щоб вирішити принцип масової адаптації до компанії, пропонується використовувати систему рекомендацій. Ця система допомагає рекомендувати відповідні продукти для клієнтів та допомагає клієнтам приймати рішення під час процесу покупки. Система рекомендувачів може працювати через провідних продавців на веб-сайті, демографічні показники споживачів або поведінку покупців. Проте існують 3 основних способи рекомендацій: рекомендації продукту безпосередньо клієнтам, надання детальної інформації про продукти та виявлення думок або критичних відгуків інших покупців. Це користь для споживчого досвіду без фізичного шопінгу. Загалом, система рекомендацій використовується для зв'язку з клієнтами в Інтернеті та допомагає знаходити потрібні продукти, які вони хочуть ефективно та безпосередньо.

Електронна комерція забезпечує зручність для клієнтів, оскільки їм не потрібно виходити з дому, і лише потрібно переглядати веб-сайт в Інтернеті, особливо для покупки продуктів, які не продаються в магазинах поруч. Це може допомогти клієнтам придбати більш широкий асортимент товарів і заощадити час покупців. Споживачі також отримують владу через інтернет-магазини. Вони здатні досліджувати товари та порівнювати ціни серед роздрібних продавців. Крім того, онлайн-покупками часто передбачено код стимулювання збуту або знижок, отже, більша ціна для клієнтів. Крім того, електронна комерція забезпечує детальну інформацію про продукти; навіть працівники магазину не можуть запропонувати таке детальне пояснення. Клієнти також можуть переглядати та відслідковувати історію замовлень в Інтернеті.

Для того, щоб отримати список підготовчих інструкцій для Sku та вхідні настанови, то потрібно викликати метод GetInboundGuidanceAndPrepInstructions-

ForSKU передавши йому екземпляр класу ShipmentSetting. Результатом методу буде об’єкт класуInboundGuidanceAndPrepInstructionsModel.Клас призначений для опису вхідних та підготовчих інструкцій для продукту поточної посилки.

Використовуючи цю перевагу можна прокащити сканування товарів під час формування пакунків з товарами. В свою чергу таким чином можна зекономити час, що міг бути витрачений на пошук товару використовуючи клавіатуру і встановлення кількості товарів для пакунку.

2.3 Автоматизація створення посилок перед їх обробкою

Процес автоматизації заключається в тому, що користувач створюючи посилку може записати дані про кожний пакунок та його вміст до бази даних. І потім опрацьовувати та виводити цю інформацію в більш зручному вигляді, який буде ідеально підходити для сканування за допомогою штрих-кодів.

Користувач вибирає товари та їх кількість, які він бажає включити до поточної посилки також вказує кількість товарів на рівні одного пакунку. Користувач визначає цю кількість базуючись на розмірах коробки. Якщо в одну коробку може за розмірами вміститись 8 одиниць товару, а користувач бажає відправити 100 одиниць, то буде вираховуватись оптимальна кількість. Оптимальною кількістю називається така кількість, при якій буде використано максимально можливо місця в коробці. Тобто, ціла частина ділення числа 100 і 8 буде 12 і 4 піде в остачу. Таким чином для пакування 100 одиниць товару потрібно 12 коробок, в кожній коробці буде міститись 8 одиниць товару. Математичними обрахунками кількість виходить 96 предметів. Далі йде вирішення для останніх 4-х предметів. Алгоритм побудований таким чином, якщо користувач в наявності має достатню кількість, щоб доповнити невистачаючу кількість, то це значення округлюється до 8 (кількість на рівні пакунку). Інакше остача відкидується і створюється пакунок з кількістю без остачі. Розглядаючи цей приклад – це число 96. Тобто формується посилка на 96 предметів, в якій буде 12 пакунків по 8 одиниць товару в кожному пакунку. Така процедура обраховується для кожної кількості товару.

Найважливішими чинниками, що визначають, чи повертаються клієнти на веб-сайт, є простота використання та наявність зручних для користувача функцій. Тестування юзабіліті є важливим для пошуку проблем та вдосконалень на веб-сайті. Методи оцінки юзабіліті включають евристичну оцінку, когнітивне керівництво та тестування користувачів. Кожен метод має свої особливості та підкреслює різні аспекти роботи користувача.

Інтернет-магазини, як правило, доступні 24 години на добу, і багато споживачів в країнах Заходу мають доступ до Інтернету як на роботі, так і вдома. Інші установи, такі як Інтернет-кафе, громадські центри та школи, також забезпечують доступ до Інтернету. На відміну від відвідування звичайного магазину роздрібної торгівлі потрібні поїздки чи оплачувати проїзд, заправлення паливом, стоянка або автобусні квитки, і, як правило, повинні проходити в робочі години. Доставка завжди була проблемою, яка вплинула на зручність онлайн-шопінгу. Проте, щоб подолати це, багато продавців принесли сервісне обслуговування магазину. Тепер це означало, що клієнти можуть придбати товари в Інтернеті та забирати їх у сусідньому магазині, що робить покупки в Інтернеті більш вигідними для клієнтів. У випадку виникнення проблеми з товаром (наприклад, продукт не був замовлений споживачем або продукт не був задовільним), споживачі стурбовані легкістю повернення товару в обмін на правильний товар або відшкодування. Споживачам, можливо, доведеться звернутися до роздрібного продавця, відвідати поштовий офіс та оплатити доставку, а потім чекати заміну або відшкодування. Деякі інтернет-компанії мають більш щедру політику повернення, щоб компенсувати традиційну перевагу фізичних магазинів. Наприклад, Інтернет-магазин роздрібної торгівлі взуттям Zappos.com містить ярлики для безкоштовної доставки відправлення, і не стягує плату за поновлення, навіть для прибутків, які не є результатом помилки продавця.

Враховуючи відсутність можливості перевіряти товар перед покупкою, споживачі піддаються підвищеному ризику шахрайства, ніж очні операції. При замовленні товару в Інтернеті елемент може не працювати належним чином, у

нього можуть бути дефекти, або це може не бути тим самим предметом, зображеним на онлайн-фотографії. Купці також ризикують сфальсифікувати покупки, якщо клієнти використовують вкрадені кредитні картки або шахрайську відмову від покупки в Інтернеті. Проте, продавці стикаються з меншим ризиком фізичної крадіжки, використовуючи склад, а не торговий візок. Шифрування Secure Sockets Layer (SSL) загалом вирішило проблему перехоплення номерів кредитних карток між споживачем та продавцем. Проте треба довіряти продавцеві (і працівникам) не використовувати інформацію про кредитну картку згодом для власних покупок, а не передавати інформацію іншим. Крім того, хакери можуть втрутитися на веб-сайт продавця та викрадати імена, адреси та номери кредитних карт, хоча Стандарт безпеки даних про обробку платіжних карток призначений для мінімізації наслідків таких порушень.

Крадіжка особистих даних все ще залишається проблемою для споживачів. У 2000-х роках ряд високооплачуваних перерв підказав деяким штатам США вимагати розкриття інформації споживачам, коли це станеться. Комп'ютерна безпека стала головною проблемою для торговців та постачальників послуг електронної комерції, які застосовують контрзаходи, такі як брандмауери та антивірусне програмне забезпечення для захисту своїх мереж.

Фішинг - це ще одна небезпека, коли споживачі змушені думати, що вони мають справу з авторитетним продавцем, коли вони фактично маніпулювалися передачею приватної інформації в систему, що працює від зловмисної партії.

Далі ці дані передаються на обробку до Amazon. Сервери Amazonобробляють запит і дають у відповідь інформацію про виконавчі центри до яких потрібно відправити певні товари.

Наприклад, до першого центру слід відправити 50 одиниць товару, до другого – 30, третього – 20. При цьому в поточного товару кількість одиниць на коробку дорівнює 5. Слідує те, що до першого центру буде відправлено 10 коробок, другого – 6, третього 4. Те саме Amazon пропонує і для інших товарів.

Обробка посилки заключається в тому, що потрібно відсканувати товари необхідні для відправки, наклеєти на них наліпки та скласти у пакунки. Після

цього пакунки закривають і наліплюють наклейку про вміст пакунку, а пакунок кладуть на палет, який призначений для окремого виконавчого центру. Про це все користувач інформується в процесі обробки товарів.

Спочатку користувач заповнює блок товарів, для яких необхідно створити поточну посилку. Користувач може це зробити безпосередньо на сторінці або попередньо добавити їх за допомогою сторінки Inventory. Далі користувач налаштовує відомості про посилку та відправника (його контактну інформацію) і після цього натискає кнопку Next.

Сервер з’єднується з серверами Amazonта передає усю необхідну інформацію та отримує відповідь, яка буде записана до локальної бази даних та буде використовуватись протягом роботи з посилкою на сторінці Inbound FBA Shipments.

Електронна комерція забезпечує зручність для клієнтів, оскільки їм не потрібно виходити з дому, і лише потрібно переглядати веб-сайт в Інтернеті, особливо для покупки продуктів, які не продаються в магазинах поруч. Це може допомогти клієнтам придбати більш широкий асортимент товарів і заощадити час покупців. Споживачі також отримують владу через інтернет-магазини. Вони здатні досліджувати товари та порівнювати ціни серед роздрібних продавців.

Крім того, онлайн-покупками часто передбачено код стимулювання збуту або знижок, отже, більша ціна для клієнтів. Крім того, електронна комерція забезпечує детальну інформацію про продукти; навіть працівники магазину не можуть запропонувати таке детальне пояснення. Клієнти також можуть переглядати та відслідковувати історію замовлень в Інтернеті. Тим не менш, електронна комерція не має взаємодії з людьми для клієнтів, особливо тих, хто віддає перевагу особистому зв'язку.

Клієнти також стурбовані безпекою онлайн-транзакцій і, як правило, залишаються лояльними до відомих роздрібних торговців. В останні роки роздрібні торговці одягом, такі як Томмі Хілфігер, почали додавати платформи Virtual Fit на свої сайти електронної комерції, щоб зменшити ризик того, що клієнти купують одяг неправильного розміру, хоча ці варіанти значно

відрізняються за своїм призначенням.

Коли замовник шкодує про покупку продукту, це передбачає повернення товару та відшкодування. Цей процес незручно, тому що клієнтам потрібно упакувати та відправити товар. Якщо продукти дорогі, великі або тендітні, це стосується питань безпеки.

Після завершення сканування товарів та формування пакунків йде підтвердження готовності до відправки. Процес полягає в тому, що Amazonмає бути проінформований про те, що пакунки готові та будуть відправленні у найближчому часі. У відповідь Amazon повертає наліпки, які потрібно буде наклеєти на контейнери, це так звані транспортні наліпки. На них вказується інформація про місце відправлення та місце призначення контейнерів. На цьому процес вважається завершеним. В загальному такий процес потребує 36 дій оператора. Дані були отримані в результаті моделювання контрольного прикладу.

Найважливішими чинниками, що визначають, чи повертаються клієнти на веб-сайт, є простота використання та наявність зручних для користувача функцій. Тестування юзабіліті є важливим для пошуку проблем та вдосконалень на веб-сайті. Методи оцінки юзабіліті включають евристичну оцінку, когнітивне керівництво та тестування користувачів. Кожен метод має свої особливості та підкреслює різні аспекти роботи користувача.

Інтернет-магазини, як правило, доступні 24 години на добу, і багато споживачів в країнах Заходу мають доступ до Інтернету як на роботі, так і вдома. Інші установи, такі як Інтернет-кафе, громадські центри та школи, також забезпечують доступ до Інтернету. На відміну від відвідування звичайного магазину роздрібної торгівлі потрібні поїздки чи оплачувати проїзд, заправлення паливом, стоянка або автобусні квитки, і, як правило, повинні проходити в робочі години. Доставка завжди була проблемою, яка вплинула на зручність онлайн-шопінгу.

Проте, щоб подолати це, багато продавців принесли сервісне обслуговування магазину. Тепер це означало, що клієнти можуть придбати товари в Інтернеті та забирати їх у сусідньому магазині, що робить покупки в Інтернеті

більш вигідними для клієнтів. У випадку виникнення проблеми з товаром (наприклад, продукт не був замовлений споживачем або продукт не був задовільним), споживачі стурбовані легкістю повернення товару в обмін на правильний товар або відшкодування.

Споживачам, можливо, доведеться звернутися до роздрібного продавця, відвідати поштовий офіс та оплатити доставку, а потім чекати заміну або відшкодування. Деякі інтернет-компанії мають більш щедру політику повернення, щоб компенсувати традиційну перевагу фізичних магазинів. Наприклад, Інтернет-магазин роздрібної торгівлі взуттям Zappos.com містить ярлики для безкоштовної доставки відправлення, і не стягує плату за поновлення, навіть для прибутків, які не є результатом помилки продавця.

У цьому розділі було розроблено метод автономного створення посилок, покращенно сканування товарів використовуючи штрих-коди і автоматизовано процес створення посилок перед їх відправленням, що дозволило скоротити приблизно на 39 кількість дій оператора.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ОБРОБКИ І ФОРМУВАННЯ ЗАПИТІВ НА ПОСТАВКУ ТОВАРІВ

Програмний засіб складається з трьох функціональних сторінок: Inventory, Prepare FBA Shipment, Inbound FBA Shipments. Кожна сторінка відповідає за окрему частину роботи програми.

На сторінці Inventory користувач може переглянути список товарів, які присутні у його лістингу на платформі Amazon. Товари мають детальну інформацію та статистичну інформацію відносно продажів та прибутку протягом певного періоду часу.

Також користувач може виконувати пошук за певними критеріями, а саме по полях Sku, Asin та назва товару.

Наступною функцією сторінки є додавання товарів до нової посилки, яку він може створити на сторінці Prepare FBA Shipment.

Сторінка Prepare FBA Shipment призначення для створення нових посилок. Користувач може вказати список товарів та їх кількість, після цього натиснути кнопку Next і програма в автоматичному режимі з’єднається з серверами Amazonта створить нову посилку, отримає всі необхідні дані, які буде записано до локальної бази даних. Після створення посилки користувача буде перенаправлено до сторінки Inbound FBA Shipments.

Inbound FBA Shipmentsпризначена для обробки посилок. Тобто складання товарів у коробки, клеєння наліпок на товари та коробки, пакування коробок у контейнери, а також друк наліпок для контейнера і в подальшому відправлення контейнерів у певні виконавчі центри Amazon.

3.1 Розробка інтерфейсу взаємодії «Inventory»

Сторінка Inventory має зручний інтерфейс. Ця сторінка складається з декількох елементів керування та таблиці-грід. На сторінці можливий пошук за певним критерієм, аналіз стану ринку та рівня продажів певного товару, а також можоивість додавання предметів до нової посилки. Інтерфейс сторінки зображено на рисунку 3.1.

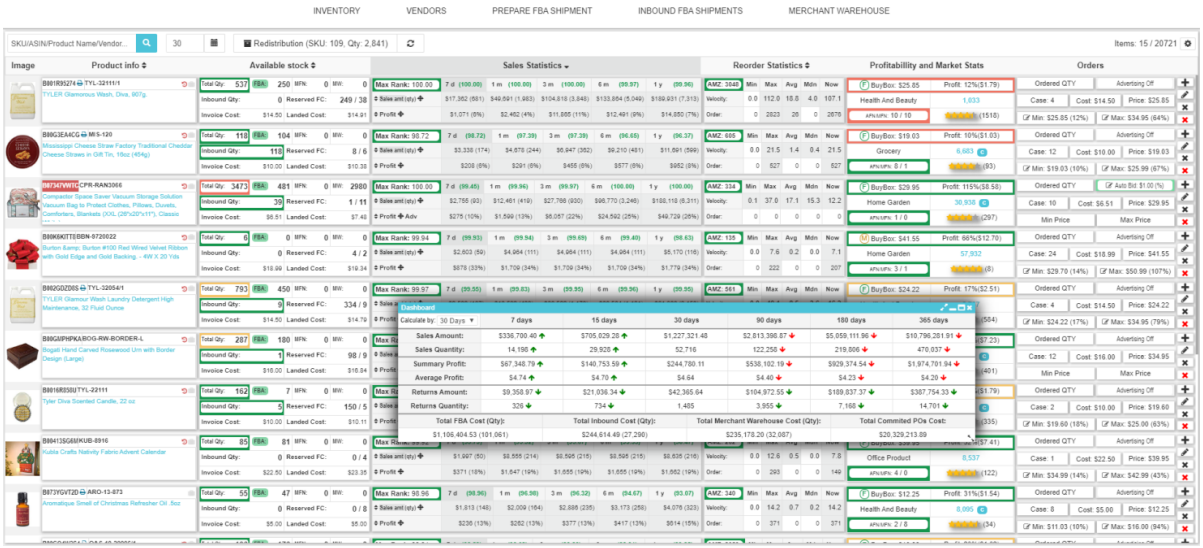

Рис. 3.1 – інтерфейс сторінки Inventory

Inventory поділяється на 3 групи класів: інформація, управління, посилка.

3.1.1 Розробка групи класів «інформація»

Група класів «інформація» відповідає за інформативну складову кожного товару. Тобто, мається на увазі опис товару, інформація про продажі та поточний стан ринку.

Основним класом є InventoryItem, який агрегує у собі наступні класи:

– Image – клас, що відповідає за інформацію про картинку для товару;
– ProductInfo – клас, що відповідає за інформацію про Aasin, Sku та назву;
– AvailableStock – клас, що відповідає за поточну кількість товарів у виконавчих центрах Amazon;
– Purchasing – клас, що містить інформацію про закупівельну ціну товару;
– PurchaseHistory – клас, що містить інформацію про історію замовлень товару у постачальника;
– SalesStatistic – клас, що містить інформацію про статистику продажів товару;
– MarketStats – клас, що містить інформацію про поточний стан ринку, поточну ціну товару на ринку, кількість пропозицій та конкурентність товару серед інших продавців;

– Profitabillity – клас, що містить інформацію про ефективність продажу товару.

Опис групи класів «інформація» наведено у додатку Б.

Для того, щоб отримати предмети, потрібно викликати метод GetItems контроллера Inventory. Метод приймає наступні параметри:

– string searchType – тип пошуку;

– string key – фільтр пошуку;

– int skip – кількість товарів, яку потрібно пропустити;

– int take – кількість товарів, яку потрібно включити до результату.

Код методу є наступним:

```
public async Task<JsonResult> GetItems(int take, int skip, string key, string sorting, string searchType)
    {
      try
      {
         var getter = new InventoryItemsGetter(new CacheInventoryItemsGetMethod(), _cacheService, MwsLoginModel);
         var response = await getter.Get(searchType, key, sorting, skip, take, _cacheService, MwsLoginModel);

         var json = Json(new { Success = true, response.Items, response.TotalItems, response.IsEditable }, JsonRequestBehavior.AllowGet);
         json.MaxJsonLength = int.MaxValue;
         return json;
      }
      catch (AggregateException ex)
      {
         foreach (var e in ex.InnerExceptions)
         {
```

```
                Extensions.NotificationExtensions.AddLog(e,
$"Inventory/GetItems({searchType}, {key}, {sorting}, {skip}, {take})");
            }
            return Json(new { Success = false, ex.InnerException.Message },
JsonRequestBehavior.AllowGet);
        }
        catch (Exception ex)
        {
            Extensions.NotificationExtensions.AddLog(ex,
$"Inventory/GetItems({searchType}, {key}, {sorting}, {skip}, {take})");
            return Json(new { Success = false, ex.Message },
JsonRequestBehavior.AllowGet);
        }
```

3.1.2 Розробка групи класів «управління»

До функцій управління товарами є:

– видалення товару;
– зміна виконавчого методу;
– зміна ціни (поточної, максимальної, мінімальної);
– зміна кількості одиниць товару на одну коробку.

Дана група має наступні класи: RemoveItemModel, ChangeFulfillmentMethod, ChangePrice, UpdateCaseQtyModel.

Клас RemoveItemModel має 1 член: publicstringSku {get; set;} – властивість відкритого рівня доступу, типу string, призначену для Sku товару. Екземпляр класу передається у метод RemoveFromAmazon контролера Inventory. Метод призначений для видалення товару з лістингу Amazon.

Клас ChangeFulfillmentMethod призначений для опису товару, для якого потрібно змінити метод обробки товару на платформі Amazon. Є 2 методи: MFN (merchant) та AFN (amazon).

Клас такі члени:

– public string Sku {get;set;} – властивість відкритого рівня доступу, призначена для Sku товару;
– public FulfillmentType Type {get;set;} – властивість відкритого рівня доступу, призначена для методу виконання.

Для того, щоб змінити метод виконання товару, потрібно у метод ChangeFulfillmentMethodForSku контролера Inventory передати екземпляр класу ChangeFulfillmentMethod.

Метод ChangePrice призначений для оновлення ціни товару. Метод приймає екземпляр класу ChangePriceModel, який містить наступні члени:

– public string Sku {get;set;} – властивість відкритого рівня доступу, призначена для Sku;
– public PriceType Type {get; set;} – властивість відкритого рівня доступу, призначена для вказання типу ціни (Min, Max, Current).

Код методу є наступним:

У контролері Inventory також передбачено функцію зміни одиниць товару на одну коробку – UpdateCaseQty. Метод приймає екземпляр класу UpdateCaseQtyModel, який містить 2 властивості:

– public string Sku {get;set;} – відкрита властивість типу string, призначена для Sku;
– public int CaseQty {get;set;} – відкрита властивість типу int, призначена для кількості одиниць товару.

Код методу є наступним:

```
public async Task<JsonResult> UpdateCaseQty(stringsku, intqty = 0)
   {
      try
      {
         using (var context = serviceMiningReadOnlyContext)
         {
```

```
            var data = await
context.FbaInboundShipmentItemsStatuses.FirstOrDefaultAsync(i => i.Sku == sku);
            if (data != null)
                context.Entry(data).State = EntityState.Modified;
            else
                context.FbaInboundShipmentItemsStatuses.Add(new
FbaInboundShipmentItemsStatus());
            await context.SaveChangesAsync();
            return new { Success = true };
          }
        }
        catch (Exception ex)
        {
           return new { Success = false, ex.Message };
                }
```

3.1.3 Розробка групи класів «посилка»

До групи класів «посилка» відсноситься клас AddToFbaModel. Даний клас описує предмет, що потрібно передати на сторінку PrepareFBAShipment. Клас містить наступні члени:

- public string Sku {get; set;} – властивість відкритого модифікатора доступу, типу string, призначена для Sku;
- public int Qty {get;set;} – властивість відкритого модифікатора доступу, типу int, призначена для кількості одиниць товару;
- public int CaseQty {get;set;} – властивість відкритого модифікатора доступу, типу int, призначена для кількості одиниць товару на одну коробоку.

Для того, щоб додати предмети до посилки, потрібно викликати метод AddToFba контролера Inventory, передавши йому модель класу AddToFbaModel.

3.2 Розробка інтерфейсу взаємодії «Prepare FBA Shipment»

Сторінка PrepareFBAShipmentділиться на 3 частина:

– блок для списку товарів, для яких необхідно створити посилку;

– блок для налаштування деталей посилки;

– блок для налаштування інформації про відправника.

Інтерфейс сторінки Prepare FBA Shipment зображено на рисунку 3.2.

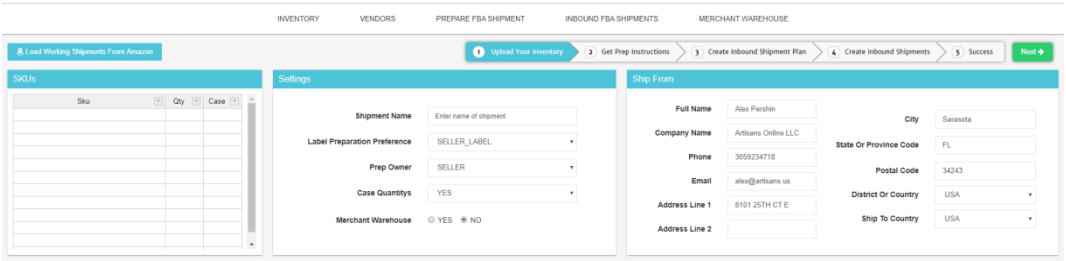

Рис. 3.2 – інтерфейс сторінки Prepare FBA Shipment

Спочатку користувач заповнює блок товарів, для яких необхідно створити поточну посилку. Користувач може це зробити безпосередньо на сторінці або попередньо добавити їх за допомогою сторінки Inventory. Далі користувач налаштовує відомості про посилку та відправника (його контактну інформацію) і після цього натискає кнопку Next.

Сервер з’єднується з серверами Amazon та передає усю необхідну інформацію та отримує відповідь, яка буде записана до локальної бази даних та буде використовуватись протягом роботи з посилкою на сторінці Inbound FBA Shipments.

3.2.1 Розробка групи класів «посилка»

До групи класів «посилка» можна віднести такий клас AddToFbaItem. Цей клас призначений для опису товару, який необхідно включити до посилки, яку буде створено. Клас містить наступні члени:

– public string Sku {get; set;} – властивість типу string, з публічним модифікатором доступу, призначена для Sku;

– public int Qty {get; set;} – властивість типу int, з публічним

модифікатором доступу, призначена для кількості одиниць товару;

– public int CaseQty {get; set;} – властивість типу int, з публічним модифікатором доступу, призначена для кількості одиниць товару на одну коробку.

3.2.2 Розробка групи класів «створення»

До даної групи можна віднести класи: ShipmentSetting, InboundShipment Plan, InboundGuidanceAndPrepInstructionsModel.

Клас ShipmentSetting призначений для опису налаштувань поточної посилки. Даний клас містить наступні члени:

– public string ShipmentName {get;set;} – властивість з публічним модифікатором доступу, типу string, призначений для назви посилки;
– public List<AddToFbaItem> Items {get;set;} – властивість з публічним модифікатором доступу, типу List<AddToFbaItem> (список AddToFbaItem), призначений для предметів, які потрібно включити до посилки.

Для того, щоб отримати список підготовчих інструкцій для Sku та вхідні настанови, то потрібно викликати метод GetInboundGuidanceAndPrepInstructions-ForSKU передавши йому екземпляр класу ShipmentSetting. Результатом методу буде об’єкт класуInboundGuidanceAndPrepInstructionsModel.Клас призначений для опису вхідних та підготовчих інструкцій для продукту поточної посилки. Він має наступні члени класу:

– public string SellerSKU {get;set;} – властивість типу string, що має публічний рівень доступу та призначена для опису SKU;
– public string ASIN {get;set;} – властивість типу string, що має публічний рівень доступу та призначена для опису ASIN;
– public string InboundGuidance {get;set;} – властивість типу string, що має публічний рівень доступу та призначена для опису вхідних інструкцій продукту;
– public string PrepInstructionList {get;set;} – властивість типу string, що має

публічний рівень доступу та призначена для опису підготовчих інструкцій продукту;

– public int Qty {get;set;} – властивість публічного рівня доступу, типу int, призначена для кількості товару;
– public int CaseQty {get;set;} – властивість публічного рівня доступу, типу int, призначена для кількості одиниць товару для однієї коробки.

Код методу є наступним:

```
public async Task<ActionResult>
GetInboundGuidanceAndPrepInstructionsForSKU(ShipmentSetting Setting)
   {
      var result =
mwsService.GetInboundGuidanceAndPrepInstructionsForSKU(Setting);
      SellerSKUList SellerSKULists = new SellerSKUList();
      Setting.ShipmentItems = new List<ShipmentItem>();
      foreach (var item in Items)
      {
         var i = Setting.ShipmentItems.SingleOrDefault(w => w.SKU == item.SKU);
         if (i != null)
         {
            i.Quantity += item.Quantity;
            continue;
         }
         Setting.ShipmentItems.Add(item);
      }
      SellerSKULists.Id.AddRange(Setting.ShipmentItems.Select(w => w.SKU));
      return Json(new
      {
         Success = true,
         Message = result,
```

```
        Model = SellerSKUList,
    }, JsonRequestBehavior.AllowGet);
        }
```

Клас InboundShipmentPlan описує модель посилки, що буде передано серверам Amazon, та на основі неї буде створено посилку в системі Amazon. Результатом функції є інформація про виконавчі центри та коробки, які потрібно створити протягом процесу обробки посилки.

Клас має наступні члени:

– public string ShipmentId {get;set;} – ідентифікатор посилки;

– public string DestinationFulfillmentCenterId {get; set;} – ідентифікатор виконавчого центру;

– public List<ShipmemtItems> Items {get;set;} – предмети, що відносяться до поточної посилки.

Для того, щоб створити посилку, потрібно викликати метод CreateInbound ShipmentPlan, що знаходиться в контролері FbaShipPrep, передавши йому аргумент типу InboundShipmentPlan.

Код методу є наступним:

```
public async Task<ActionResult>
CreateInboundShipmentPlan(InboundShipmentPlanplan)
  {
    Guid newDbCollectionId = Guid.Empty;
    List<ShipmentItem> SkusQty = new List<ShipmentItem>();
    List<dynamic> plans = new List<dynamic>();
    foreach (var planByKey in plansByLoadType)
    {
      plans.AddRange(item.Items.member.Select(w => new
      {
        ShipmentId = item.ShipmentId,
        DestinationFulfillmentCenterId = item.DestinationFulfillmentCenterId,
```

```
            LabelPrepType = item.LabelPrepType,
            Sku = w.SellerSKU,
            Quantity = w.Quantity
        }));
    }
    var missItems = Setting.ShipmentItems.Where(s => !SkusQty.Any(x => x.SKU ==
s.SKU)).ToList();
    return Json(new
    {
        Success = true,
        Message = newDbCollectionId,
        Plans = plans,
    }, JsonRequestBehavior.AllowGet);
}
```

3.3 Розробка інтерфейсу взаємодії «Inbound FBA Shipments»

Сторінка Inbound FBA Shipments призначена для перегляду та обробкиіснуючих посилок. Тут користувач може сканувати предмети за допомогою штрих-кодів. Формувати вміст коробки, клеєти на неї наліпки, а після цього складати все на палети або в контейнери, які в свою чергу теж мають містити інформацію про виконавчий центр, до якого потрібно відправити товар на реалізацію.

Інтерфейс сторінки зображений на рисунку 3.3.

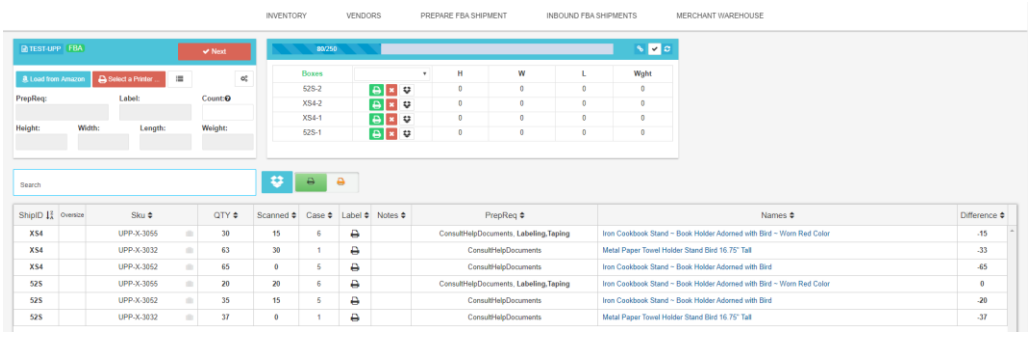

Рис. 3.3 – інтерфейс сторінки Inbound FBA Shipments.

Класи, які створені для роботи з цією сторінкою ділятись на 3 групи:

– група класів «колекції»;

– група класів «управління пакунками»;

– група класів «сканування»;

– група класів відправлення.

Розглянемо кожну групу класів окремо.

3.3.1 Розробка групи класів «колекції»

До даної групи відноситься клас FbaShipmentCollection, який має наступні властивості:

– public Guid Id { get; set; } – властивість публічного рівня доступу, призначена для ідентифікатора колекції;

– public string Name { get; set; } – властивість, яка призначена для назви колекції та має публічний рівень доступу;

– public DateTime UpdateDate { get; set; } – властивість типу DateTime, яка має публічний рівень доступу та призначена для збереження дати останнього оновлення колекції;

– public DateTime CreateDate { get; set; } – властивість типу DateTime, яка має публічний рівень доступу та призначена для збереження дати створення колекції.

Для того, щоб отримати список усіх колекцій (посилок) потрібно викликати метод GetAllShipmentCollections, контролера FbaInbound. Метод приймає наступні аргументи:

– int page – агрумент типуint, призначений для передавання поточної сторінки;

– int pageSize – аргумент типу int, призначений для передавання кількості записів на одну сторінку;

– string nameFilter – аргумент типу string, призначений для фільтрації колекцій за їх іменем;

– string skuFilter – аргумент типу string, призначений для фільтрації

колекцій за Sku, що присутні у колекції.

Код методу є наступним:

```
public async Task<ActionResult> GetAllShipmentCollections(int page, int pageSize, string nameFilter = null, string skuFilter = null)
        {
            var allDbFbaInboundShipmentCollections = _sellermogulContext.FbaShipmentCollections.Where(c => c.CompanyId == UserInfo.CompanyId);
            allDbFbaInboundShipmentCollections = allDbFbaInboundShipmentCollections.Where(c => c.Name.Contains(nameFilter));
            allDbFbaInboundShipmentCollections = allDbFbaInboundShipmentCollections.Where(c => _sellermogulContext.FbaInboundShipments.Any(s => s.FbaShipmentCollectionId == c.Id && _sellermogulContext.FbaInboundShipmentItems.Any(i => i.FbaInboundShipmentId == s.Id && i.Sku.Contains(skuFilter.ToUpper()))));
            allDbFbaInboundShipmentCollections = allDbFbaInboundShipmentCollections.OrderByDescending(c => c.Date);
            var dbFbaInboundShipmentCollections = await allDbFbaInboundShipmentCollections.Skip((page - 1) * pageSize).Take(pageSize);
            return Json(new
            {
               Success = true,
               TotalData = allDbFbaInboundShipmentCollections.Count(),
               Message = dbFbaInboundShipmentCollections
            }, JsonRequestBehavior.AllowGet);
        }
```

Для того, щоб отримати інформацію для певної колекції, потрібно викликати метод GetShipmentCollectionData контролера FbaInbound. Метод приймає на вході аргумент collectionId типу Guid, а у відповідь отримує модель типу ShipmentCollectionData.

Клас ShipmentCollectionData має наступні члени:

– public List<Shipment> Shipments {get;set;} – властивість типу List<Shipment> (список посилок), призначена для інформації про посилки у колекції;
– public List<Box> Boxes {get;set;} – властивість типу List<Box> (список коробок), призначена для інформації про коробки у колекції;
– public List<DestinationAddress> Destinations {get;set;} – властивість типу List<DestinationAddress> (список виконавчих центрів), призначена для інформації про виконавчі центри Amazon призначені для посилок поточної колекції.

3.3.2 Розробка процесу «управління пакунками»

До групи класів «управління пакунками» відноситься клас Box.Він містить у собі всю інформацію про коробку. Клас Box має наступні члени:

– public Guid Id {get; set;} – ідентифікаторкоробки;
– public string BoxNumber {get;set;} – номер коробки;
– public Guid ShipmentId {get; set;} – ідентифікатор посилки, до якої відноситься коробка;
– public double Width {get; set;} – ширина коробки;
– public double Height {get; set;} – висота коробки;
– public double Length {get; set;} – довжина коробки;
– public double Weight {get; set;} – вага коробки.

Для управління коробками введено 2 методи:

– AddOrUpdateBox – метод, призначений для створення коробки або оновлення існуючої;
– DeleteBox – метод, призначений для видалення існуючої коробки.

Для того, щоб створити коробку або оновити існуючу потрібно передати модель Box в метод AddOrUpdateBox контролера FbaInbound.

Код методу є наступним:

```
public async Task<JsonResult> AddOrUpdateBox(Box model)
```

```
{
    var allParcels =
        await _sellermogulContext.FbaInboundParcels.
        Where(p => p.CollectionId == model.CollectionId && p.ShipmentId ==
model.ShipmentId).
        OrderBy(p => p.ParcelIndex).
        Select(p => (int)p.ParcelIndex).ToListAsync();
    _sellermogulContext.FbaInboundParcels.AddOrUpdate(model);
    await _sellermogulContext.SaveChangesAsync();
    return Json(new { Success = true, Message = model },
JsonRequestBehavior.AllowGet);
        }
```

Для того, щоб видалити існуючу коробку, потрібно викликати метод DeleteBox, контролера FbaInbound, передавши йому модель Box.

Код методу є наступним:

```
public async Task<JsonResult> DeleteBox(FbaInboundParcel model)
{
    var dbModel = await
_sellermogulContext.FbaInboundParcels.FirstOrDefaultAsync(p => p.Id == model.Id);
    _sellermogulContext.FbaInboundParcels.Remove(dbModel);
    await _sellermogulContext.SaveChangesAsync();
    return Json(new { Success = true, Message = "Removed" },
JsonRequestBehavior.AllowGet);
        }
```

3.3.3 Розробка процесу «сканування»

До даної групи відноситься клас ParcelItem. Клас має наступні властивості:

- publicGuidId {get; set;} – ідентифікатор предмету;
- public string Sku {get; set;} – Sku товару;
- public int Scanned {get; set;} – кількість відсканованих предметів;

– public Guid BoxId {get; set;} – ідентифікатор коробки, до якої відноситься предмет.

Для роботи з класом ParcelItem додано 2 метода управління:

– AddOrUpdateBoxItem – метод, призначений для додавання товару до коробки або оновлення інформації про нього;
– DeleteBoxItem – метод, призначений для видалення товару з коробки.

Для того, щоб додати предмет до коробки або оновити його, потрібно викликати метод AddOrUpdateBoxItem контролера FbaInbound, передавши йому аргументом екземпляр класу ParcelItem.

3.3.4 Розробка процесу «відправлення»

Для того, щоб повідомити Amazon про те, що посилки готові до відправлення, потрібно викликати метод VoidTransportRequest контролера FbaInbound передавши аргументом ідентифікатор колекції (значення типу Guid).

Код методу є наступним:

```
public JsonResult VoidTransportRequest(Guid shipmentCollectionId, string
shipmentId)
   {
      var shipmentIds = _sellermogulContext.FbaInboundShipments.Where(s =>
s.FbaShipmentCollectionId == shipmentCollectionId && s.Id == shipmentId).Select(s
=> s.Id).ToArray();
      var dbShipmentsList =
_sellermogulContext.FbaInboundShipments.SqlQuery("Select * From
FbaInboundShipments " + SqlExtensions.SqlInOperation(shipmentIds, "Id")).ToList();
      foreach (var _shipmentId in shipmentIds)
      {
         var voidTransportRequest =
fbaMws.InvokeVoidTransportRequest(_shipmentId);
         var shipment = dbShipmentsList.FirstOrDefault(s => s.Id == _shipmentId);
         shipment.TransportStatus = (int)TransportStatuses.Voided;
```

```
            _sellermogulContext.FbaInboundShipments.AddOrUpdate(shipment);
        }
        _sellermogulContext.SaveChanges();
        return Json(new { Success = true, Message = "Ok" },
JsonRequestBehavior.AllowGet);
            }
```

Після виконання цього методу посилка є готовою до відправки в Amazon.

3.4 Розробка процесу друку наклейок

Паралельно з викликами AddOrUpdateBox, AddOrUpdateBoxItem та VoidTransportRequest викликається метод PrintLabels, який виконує друк наліпок. Метод приймає екземпляр класу PrintLabelModel. Клас має такі члени:

- public int Count {get; set;} – кількість наліпок;
- public LabelType Type{get; set;} – тип наліпки;
- Dictionary<string, string> Params {get;set;} – параметри наліпки.

Код методу є наступним:

```
public void PrintLabel(PrintLabelModelprintData) {
            if (printData.dataType == "PrintParams") {
                var labelType = GetLabelTypeFromString(printData.Type);
                var labelId = (int)labelType;
                if (!_labelsSettings.ContainsKey(labelId))
                {
                    ShowErrorDialog("You must configure " + labelType.ToString() + "
label.");
                    return;
                }
                var priterSettingsModel = _labelsSettings[labelId];
try
                {
                    if (priterSettingsModel.UseZPL)
```

```
                {
                    var printerZplTemplate = GenerateLabelZpl(printData.Params);
                    foreach (var page in printerZplTemplate)
                    {

RawPrinterHelper.SendStringToPrinter(priterSettingsModel.PrinterName, page);
                    }
                }
                else
                {
                    var printPDFModel = GenerateLabelPdf(printData.Params);
                    if (null != printPDFModel)
                        PrintPdfDocument(printPDFModel, labelType, printData.Count);
                }
            }
            catch (Exception ex)
            {
                ShowErrorDialog(ex.Message);
}           }       }
```

У даному розділі розроблено програмний засіб для автономного формування замовлень на відправку до Amazon. Створено зручний і зрозумілий користувацький інтерфейс. Програмний засіб написано мовами C# та JavaScript у середовищі Visual Studio 2019. Розроблено інтерфейс та блок класів для сторінки "Inventory". Описано процес фільтрування товарів за певними критеріями з використанням розробленого програмного засобу. Розроблено інтерфейс та блок класів для сторінки "Prepare FBA Shipment". Запропоновано послідовність дій створення замовлення для торгової платформи "Amazon" на основі штрих-кодів. Розроблено інтерфейс та блок класів для сторінки "Inbound FBA Shipments". Описано процес автоматичної обробки замовлень з використанням штрих-кодів.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ

У сучасних умовах відсутність впровадження інновацій відчувається особливо гостро. Підприємства потребують швидкого впровадження досягнень науково-технічного прогресу галузі, а це звісно вимагає зменшення часу на проведення науково-дослідних робіт і скорочення строку окупності витрат. Виходячи з цього, доцільно орієнтуватися на час проведення науково-дослідних робіт та розробку експериментального зразка продукту не більше 1 року, при чому технічні показники результатів плануються на рівні кращих світових зразків; термін окупності витрат у межах 1-2 років і менше.Впродовж подальшого розвитку роботи витрати повинні поступово зменшуватися, це свідчитиме про успішну реалізацію продукту і його своєчасне удосконалення.

На основі економічних розрахунків можна довести економічну доцільність та ефективність впровадження отриманих результатів виконаних науково-дослідних робіт у виробництво, тобто здійснити так звану комерціалізацію наукових розробок.

Саме цим завданням присвячено даний розділ магістерської кваліфікаційної роботи і передбачає він виконання таких етапів робіт (рис. 4.1):

Рисунок 4.1 – Складові економічної частини магістерської кваліфікаційної роботи

Саме на такі складові буде поділено економічну частину даної магістерсь-

кої роботи. Усі подальші економічні розрахунки, будуть висвітлені у згаданих підрозділах економічної частини. У комплексі ці етапи дозволять побачити цілісну картину доцільності розробки та впровадження запропонованого рішення.

4.1 Технологічний аудит розробки

Метою проведення оцінювання комерційного потенціалу розробки є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання комерційного потенціалу розробки будемо здійснювати за 12-ма критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 - Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Багато аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 4.1

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
Практична здійсненність					
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідно регламентні документи та велика кількість дозвільних документів на виробництво	Необхідно отримання великої кількості дозвільних документів на виробництво	Процедура отримання дозвільних документів для виробництва	Необхідно тільки повідомлення відповідним	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі складеної таблиці ряд незалежних експертів, у нашому випадку керівник магістерської роботи та викладачі випускової кафедри виставили різні бали. Результати цього оцінювання комерційного потенціалу внесено до таблиці 4.2.

Таблиця 4.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 Очкуров М.А., к.т.н., доц. кафедри ОТ	2 Колесник І. С., к.т.н., доц. кафедри ОТ	3 Богомолов С.В., к.т.н., доц. кафедри ОТ
	Бали, виставлені експертами:		
1	3	3	3
2	2	2	2
3	2	3	2
4	3	2	2
5	3	2	2
6	3	4	3
7	0	1	0
8	3	3	3
9	1	1	1
10	4	4	4
11	3	4	3
12	3	3	3
Сума балів	$СБ_1 = 30$	$СБ_1 = 32$	$СБ_1 = 28$
Середньо-арифметична сума балів $\overline{Сі}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{30+32+28}{3} = \frac{90}{3} = 30$		

За даними таблиці 4.2, а також згідно рекомендацій, що наведені в таблиці 4.3, можна зробити висновок, щодо рівня комерційного потенціалу розробки.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{Сі}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 - 20	Нижче середнього
21 - 30	Середній
31 – 40	Вище середнього
41 - 48	Високий

Взявши до уваги, середньоарифметичну суму балів, $\overline{СБ} = 30$, що були виставлені експертами, можна стверджувати, що рівень комерційного потенціалу даної розробкиє середнім.

4.2 Прогнозування витрат на виконання та впровадження результатів наукової роботи

1) Основна заробітна плата розробників, які працюють над проектом визначається за формулою 4.1:

$$Зо = \frac{М}{Тр} \cdot t, (грн.) \quad (4.1)$$

де M - місячний посадовий оклад розробника;

$Тр$ - число робочих днів в місяці ($T_р$= 22дні);

t - число днів роботи розробника.

Над створенням розробки працювали керівник проекту та інженер-програміст , отже, виконаємо для них всі необхідні розрахунки :

$$Зо = \frac{9000,00}{22} \cdot 9 = 3681,81 \ (грн.)$$

$$Зо = \frac{15000,00}{22} \cdot 22 = 15000,00 \ (грн.)$$

Таблиця 4.4- Заробітна плата

Найменування посади	Місячний посадовий оклад,грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
1 Керівник	9000,00	409,09	9	3681,81
2 Інженер-програміст	15000,00	681,81	22	15000,00
Всього				18681,81

2) Додаткова заробітна плата $Зд$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми

основної заробітної плати всіх розробників та робітників, тобто:

$$Зд = (10 \dots 12\%) \cdot Зо, (грн.) \quad (4.2)$$

де $Зо$- основана заробітна плата.

$$Зд = \frac{10 \cdot 18681{,}81}{100} = 1868{,}18 (грн.)$$

3) Нарахування на заробітну плату $Нзп$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Нзп = (22\%) \cdot (Зо + Зд), (грн.) \quad (4.3)$$

$$Нзп = \frac{22 \cdot (18681{,}81 + 1868{,}18)}{100} = 4521{,}00 (грн.)$$

4) Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування $А$ в цілому були розраховані за формулою 4.4:

$$А = \frac{Ц \cdot На}{100} \cdot \frac{Т}{12}, (грн.) \quad (4.4)$$

де $Ц$ - загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

$На$ - річна норма амортизаційних відрахувань;

$Т$ - термін, використання обладнання, приміщень тощо, місяці.

Всі розрахунки зводимо до таблиці 4.5.

Таблиця 4.5 - Амортизація обладнання та приміщень

Найменування обладнання, приміщень	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
ЕОМ	12000	10%	1	1200
Приміщення для дипломного проектування	7000	2%	1	140,00
Всього				1340,00

5) Витрати на силову електроенергію B_e, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою 4.6:

$$В_е = В \cdot П \cdot Ф \cdot К_п, (грн.) \quad (4.6)$$

де В - вартість 1 кВт електроенергії, грн.;

П - установлена потужність обладнання, кВт/год;

Ф - фактична кількість годин роботи обладнання, яке задіяне на виготовлення одного виробу, годин;

K_n - коефіцієнт використання потужності, $K_n \leq 1$.

Вартість 1кВт електроенергії рівна 8,43 грн, потужність обладнання (сервера) рівна 700 Вт, що тотожно 0,7 кВт, фактична робота обладнання для роботи рівна 336 год, а коефіцієнт використання потужності приймемо за 0,8.

$$В_е = 1{,}68 \cdot 0{,}7 \cdot 336 \cdot 0{,}8 = 316{,}10 \ (грн.)$$

6) Інші витрати. Інші витрати охоплюють: загально виробничі витрати (витрати управління організацією, ремонт та експлуатація основних засобів, витрати на опалення, освітлення тощо), адміністративні витрати (проведення зборів, оплата юридичних та аудиторських послуг, тощо), витрати на збут (витрати на рекламу, перепідготовка кадрів) на інші операційні витрати (штрафи, пені, матеріальні допомоги, втрати від знецінення запасів тощо).

Інші витрати можна розрахувати за нормативами відносно основної заробітної плати основних робітників, які виготовляють продукцію, за формулою 4.7.

$$В_{ін} = Н \cdot Зо, (грн.) \quad (4.7)$$

де Н - норматив загальновиробничих витрат. Для ЕОМ Н = 230-270%.

$$В_{ін} = \frac{250 \cdot 18681,81}{100} = 46704,54 \ (грн.)$$

7) Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи - В.

$$В = 18681,81 + 1868,18 + 4521,00 + 1340,00 + 970 + 316,10 + 46704,54 = 74401,65 \ (грн.)$$

9) Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.8)$$

де β - коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Оскільки, розробка знаходиться на стадії впровадження, то $\beta \approx 0,9$;

$B_{заг}$ - загальна вартість всієї наукової роботи. У даному випадку $B_{заг} = B$.

$$ЗВ = \frac{74401,65}{0,9} = 82668,50 (грн.)$$

Отже, розрахований кошторис витрат на розробку програмного забезпечення для виділення рухомих об’єктів цифрових зображеньскладає 82668,50 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі виконано прогнозування, яку вигоду можна отримати у майбутньому від впровадження результатів даної наукової роботи.

Передбачається, що виконання наукової роботи та впровадження результатів по розробці програмного забезпечення для виділення рухомих об’єктів цифрових зображень займе 1 рік.

Основні позитивні результати від впровадження розробки очікуються протягом 3 років після її впровадження.

Саме зростання чистого прибутку забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

$$\Delta П_{\text{і}} = \sum_{1}^{n} (\Delta П_{\text{я}} \cdot N + П_{\text{я}} \Delta N)_i \, , (\text{грн.}) \qquad (4.9)$$

де $\Delta П_я$ - покращення основного якісного показника від впровадження результатів розробки у даному році;

N - основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN - покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ - основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n - кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки покращується якість програмного продукту, що дозволяє підвищити ціну його реалізації на 200 грн., а кількість потенційних користувачів ресурсу збільшиться: протягом першого року - на 720 шт., протягом другого року - ще на 610 шт., протягом третього року - ще на 565 шт.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 900 шт., а прибуток, що його отримувало підприємство на одиницю продукції до впровадження результатів наукової розробки - 500 грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства протягом наступних трьох років складе:

$$\Delta П_1 = 200 \cdot 900 + (500 + 200) \cdot 720 = 460000$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_2 = 200 \cdot 900 + (500 + 200) \cdot 610 = 320000$$

Збільшення чистого прибутку підприємства $\Delta П_i$протягом третього року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_3 = 200 \cdot 900 + (500 + 200) \cdot 565 = 250000$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахований комерційний ефект від можливого впровадження розробки ще не означає, що ця розробка реально буде впроваджена. Якщо збільшення прогнозованого прибутку від впровадження результатів наукової розробки є вигідним для підприємства, то це ще не означає, що інвестор погодиться фінансувати розробку.

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1-й крок. Розрахуємо теперішню вартість інвестицій PV, що вкладаються в

наукову розробку. Такою вартістю, можна вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану нами раніше за формулою (4.8), тобто будемо вважати, що ЗВ = PV=82668,50.

2-й крок. Для спрощення подальших розрахунків побудуємо вісь часу, на яку нанесемо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі показуються у ті терміни, коли вони здійснюються. Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 4.2.

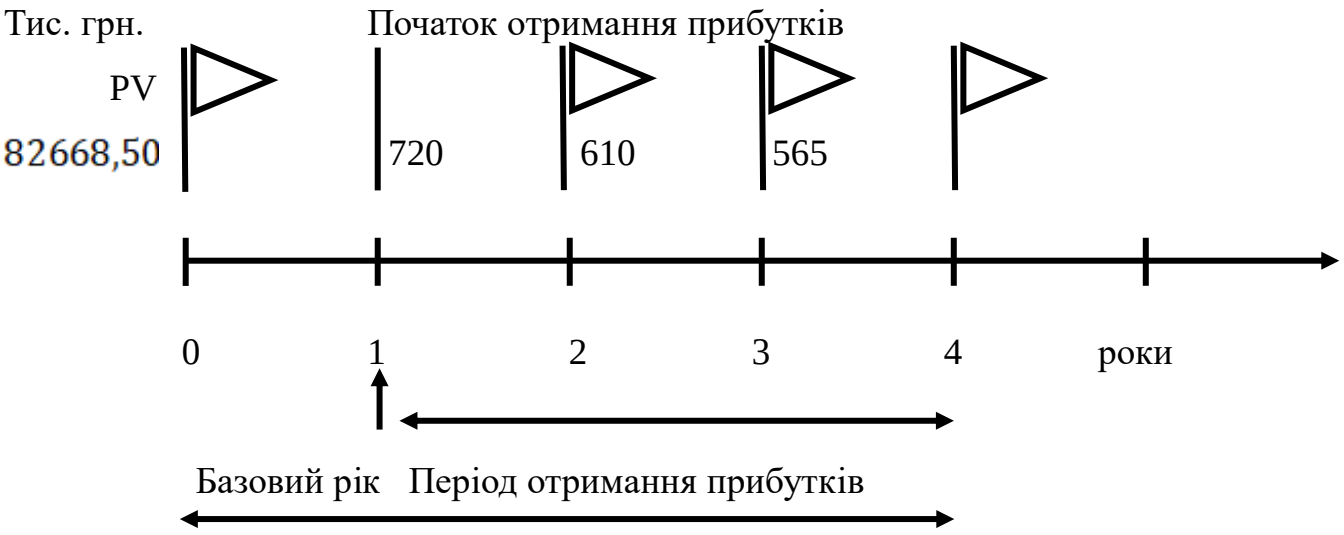

Рисунок 4.2 - Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4-й крок. Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$.

Для цього користуються формулою:

$$E_{абс} = (ПП - PV) \quad (4.10)$$

де $ПП$ - приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV - теперішня вартість інвестицій PV = ЗВ=113,995грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, (тис.\,грн.) \quad (4.11)$$

де $\Delta П_i$- збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

T - період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ - ставка дисконтування [], за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,3;

t - період часу (в роках) від моменту отримання чистого прибутку до точки «0».

Отримаємо:

$$ПП = \frac{720}{(1+0,1)^2} + \frac{610}{(1+0,1)^3} + \frac{565}{(1+0,1)^4} = 1439,245 \text{ тис. грн.}$$

Тоді $Е_{абс} = (1439,245 - 82,668) = 1356,577$ тис. грн.

Оскільки $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження може принести прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даної роботи.

5-й крок. Розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, (\%) \quad (4.12)$$

де $Е_{абс}$ - абсолютна ефективність вкладених інвестицій, грн;

PV - теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ - життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[4]{1 + \frac{1356,577}{82,668}} - 1 = \sqrt[4]{1 + 16,40} - 1 = 104\ \%$$

Далі, розрахована величина $E_в$ порівнюється з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$t = d + f, (\%) \qquad (4.13)$$

де d - середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = (0,15...0,25);

f - показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1), але може бути і значно більше.

$$t = d + f = 0,25 + 0,1 = 0,3 = 35\%$$

Величина $E_в > \tau_{мін}$, інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених коштів у реалізацію наукового проекту за формулою:

$$T_{ок} = \frac{1}{E_в}, (років) \qquad (4.14)$$

$$T_{ок} = \frac{1}{1,042} = 0,95 \text{ року.}$$

Оскільки $T_{ок}$ = 0,95 року, то фінансування розробки є доцільним.

У четвертому розділі магістерської кваліфікаційної роботи проведено розрахунки, що доводять економічну доцільність та ефективність впровадження розробленого програмного засобу. Розрахунки було поділено на 4 частини, що

утворили відповідні підрозділи даного розділу. В комплексі підрозділи дозволяють побачити картину економічної доцільності нового рішення.

В першому підрозділі виконано оцінювання комерційного потенціалу засобу. На основі думки експертів сформовано систему критеріїв та за 5-ти бальною шкалою, виставлено бали по кожному критерію. Виставлені бали, говорять про те, що рівень комерційного потенціалу є середнім.

Другий підрозділ економічної частини демонструє витрати на розробку, що розраховуються, як сума усіх статей витрат поділена на ступінь готовності продукту. Розрахований кошторис витрат на розробку складав 82668.50 грн.

Останній підрозділ висвітлює основі показники, які визначають доцільність фінансування наукової розробки певним інвестором. Такими показниками є абсолютна та відносна ефективність вкладених інвестицій, а також термін їх окупності.

Обрахована абсолютна ефективність становить 1356577 грн, що свідчить про те, що інвестор буде зацікавлений у фінансуванні даної розробки.

Відносна (щорічна) ефективність становить 104,26%, що більше мінімальної ставки дисконтування, що ще раз підтверджує зацікавленість інвестора.

Термін окупності вкладених коштів у реалізацію наукового проекту становить 0.95, що означає, що вкладені кошти повернуться через один рік.

Таким чином, можна стверджувати, що фінансування даної розробки є доцільним.

ВИСНОВКИ

У магістерській роботі було розроблено програмне забезпечення для обробки і автономного формування посилок у виконачі центри Amazon. Проаналізовано існуючі програмні рішення, виявлено, що вони мають один суттєвий недолік – незручні у користуванні, тому було розроблено метод сканування товарів за допомогою штрих-кодів.

Також було:

– проаналізовано існуючі інтернет-ресурси для віддаленої торгівлі;

– розроблено метод автономного створення посилок;

– покращено сканування товарів використвоючи штрих-коди;

– автоматизовано процес створення посилок перед їх обробкою;

– розроблено інтерфейс взаємодії користувача з програмою;

– розрахувано економічну доцільність створення програмного засобу.

У першому розділі було розглянуто існуючі програмні рішення, їх переваги та недоліки. В результаті було визначено недосконаліть існуючих рішень, які потребують більшу кількість дій оператора. Зроблено висновок, що кожен з представлених ресурсів має свої переваги та недоліки, але плюсом Amazon є те, що на нього можна повністю перекласти логістику: достатньо відправити контейнер з товаром і після цього відправляти API запити на відправку конкретного товаруклієнту, а роботи в свою чергу зроблять самі свою справу.

У другому розділі було розроблено метод автономного створення посилок, покращенно сканування товарів використовуючи штрих-коди і автоматизовано процес створення посилок перед їх відправленням, що дозволило скоротити приблизно на 39 кількість дій оператора.

У третьому розділі розроблено програмний засіб для автономного формування замовлень на відправку до Amazon. Створено зручний і зрозумілий користувацький інтерфейс. Програмний засіб написано мовами C# та JavaScript у середовищі Visual Studio 2019. Розроблено інтерфейс та блок класів для сторінки "Inventory". Описано процес фільтрування товарів за певними критеріями з

використанням розробленого програмного засобу. Розроблено інтерфейс та блок класів для сторінки "Prepare FBA Shipment". Запропоновано послідовність дій створення замовлення для торгової платформи "Amazon" на основі штрих-кодів. Розроблено інтерфейс та блок класів для сторінки "Inbound FBA Shipments". Описано процес автоматичної обробки замовлень з використанням штрих-кодів.

У четвертому розділі проведено розрахунки, що доводять економічну доцільність та ефективність впровадження розробленого програмного засобу. Розрахунки було поділено на 4 частини, що утворили відповідні підрозділи даного розділу. В комплексі підрозділи дозволяють побачити картину економічної доцільності нового рішення.

Таким чином, всі поставлені задачі були виконані, що підтверджує досягнення мети.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Методичні вказівки до виконання курсової роботи з дисципліни "Програмування" для студентів напрямів підготовки - "Комп'ютерна інженерія" та "Інформаційна безпека" / О.І. Черняк, О.М. Ткаченко.: МВ. - Вінниця, Універсум-Вінниця. – 2006

2) Onlineshopping [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Лондон : Wikipedia, 2001-2018. – Режим доступу: en.wikipedia.org/wiki/Online_shopping (дата звернення 30.03.2018) – Назва з екрана.

3) E-commerce [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Лондон : Wikipedia, 2001-2018. – Режим доступу: en.wikipedia.org/wiki/ E-commerce (дата звернення 03.04.2018) – Назва з екрана.

4) Аналоги eBay и Amazon [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Москва : E-Molotok, 2001-2018. – Режим доступу: e-molotok.com/14-auktsionnyh-ploshhadok-analogov-ebay-i-amazon-na-kotoryh-mozhno-prodavat-svoi-tovary.htm (дата звернення 03.04.2018) – Назва з екрана.

5) Основные отличия eBay и Amazon [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Москва : Partnertrade, 2001-2018. – Режим доступу: partnertrade.org/raznitsa-ebay-i-amazon (дата звернення 03.04.2018) – Назва з екрана.

6) eBay [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сан Хосе : eBay, 2001-2018. – Режим доступу: eBay.com (дата звернення 04.04.2018) – Назва з екрана.

7) Матейщук А. А. Програмний засіб для автономного формування замовлень філіалу торгової платформи Amazon.Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2019)».Збірник матеріалів.- Вінниця, ВНТУ, 2019. -с. 15-16. - [Електронний ресурс]. Режим доступу https://conferences.vntu.edu.ua/public/files/mn/mn-2019_netpub.pdf Дата звернення: листопад, 2019.

8) Amazon [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сіетл : Amazon, 1994-2018. – Режим доступу: amazon.com (дата звернення 04.04.2018) – Назва з екрана.

9) Amazon [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Лондон : Wikipedia, 2001-2018. – Режим доступу: en.wikipedia.org/wiki/Amazon_(company) (дата звернення 05.04.2018) – Назва з екрана.

10) uBid [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Флорида : uBid, 2003-2018. – Режим доступу: uBid.com (дата звернення 04.04.2018) – Назва з екрана.

11) Bonanza [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Вашингтон : Bonanza, 2015-2018. – Режим доступу: Bonanza.com (дата звернення 04.04.2018) – Назва з екрана.

12) OLA.com [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сан Хосе : OLA, 2011-2018. – Режим доступу: OLA.com (дата звернення 04.04.2018) – Назва з екрана.

13) Sellersnap [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Флорида : uBidSellersnap, 2003-2018. – Режим доступу: sellersnap.io (дата звернення 04.04.2018) – Назва з екрана.

14) Sellercloud [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Вашингтон: BonanzaSellercloud, 2015-2018. – Режим доступу: sellercloud.com (дата звернення 07.04.2018) – Назва з екрана.

15) Sellercentral [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сіетл : Amazon, 1994-2018. – Режим доступу: sellercentral.amazon.com (дата звернення 08.04.2018) – Назва з екрана.

16) AmazonMWSScratchpad [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сіетл : Amazon, 1994-2018. – Режим доступу: mws.amazonservices.com/s-cratchpad/index.html (дата звернення 09.04.2018) – Назва з екрана.

17) Албахари Д. C# 7.0. Карманный справочник / Д. Алхабари, Б. Албахари. – М. : Вильямс, 2017, - 224 с.

18) Фримен А. C# 7.0. ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов / Адам Фримен – М. : Вильямс, 2015, - 736 с.

19) Lerman J. Programming Entity Framework / Juliad Lerman – London : O'Reilly Media, 2010, - 920 с.

20) Бошемин Б. Основы ADO.NET / Боб Бошемин – М. : Вильямс, 2007, – 215 с.

21) Бондарь А. Microsoft SQL Server 2014 / Александр Бондарь – Санкт-Петербург : БХВ-Петербург, 2015, – 592 с.

22) Морето С.Bootstrap в примерах / Сильвио Морето – М. : ДМК Пресс, 2017. – 314 с.

23) Фрейн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Бен Фрейн – Санкт-Петербург : Питер, 2017, – 272 с.

24) Кантор И. Современный учебник JavaScript / Илья Кантор – СПб. : Питер, 2015, – 303 с.

25) Kozlowski P. Mastering Web Application Development With AngularJS / Pawel Kozlowski,Peter Bacon Darwin, London : Packt, 2013, 192 p.

ДОДАТОК А

Технічне завдання

Міністерство освіти і наукиУкраїни

Вінницький національний технічний університет

Інститут інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____ проф., д.т.н. Т.Б. Мартинюк

«___»_____2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Програмний засіб для автономного формування замовлень

філіалу торгової платформи Amazon

08-23.МКР.012.00.000 ТЗ

Науковий керівник: к.т.н., доцент

_____Черняк О. І.

(підпис)

Магістрант групи КІ-18м

_____Матейщук А. А.

(підпис)

Вінниця 2019 р.

1. Підстава для виконання магістерської кваліфікаційної роботи (МКР)

а) актуальність досліджень обумовлена спрощенням процесу створення посилки та її обробкою з подальшим відправленням до виконавчих центрів Amazon;

б) наказ про затвердження теми магістерської кваліфікаційної роботи.

2. Мета і призначення МКР

а) метою є розбробка програмного засобу формування запитів на логістику в Amazon з розробленим методом створення автономних посилок;

б) призначення розробки – автономне створення посилок на основі поточного стану продажів та ринку.

3. Вихідні дані для виконання МКР

Розробити метод автономного створення посилок.

4. Вимоги до виконання МКР

- розробити метод автономного створення посилок;
- покращити сканування товарів використвоючи штрих-коди;
- автоматизувати створення посилок перед їх обробкою.

5. Етапи МКР та очікувані результати

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Техніко-економічне обгрунтування	03.10.19	06.10.19	розділ 1
2	Аналіз і вибір математичних моделей оцінювання	07.10.19	17.10.19	розділ 2
3	Розробка структури та інформаційного і програмного забезпечення сайту	18.10.19	15.11.19	розділ 3
4	Розробка і тестування програмного засобу	16.11.19	25.11.19	розділ 4
5	Підготовка економічної частини	26.11.19	30.11.19	розділ 4
6	Оформлення ПЗ і презентації	01.12.19	07.12.19	ПЗ, презентація

6. Матеріали, що подаються до захисту МКР

Пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами

7. Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8. Вимоги до оформлення МКР

Вимоги викладені в «Положенні про порядок підготовки магістрів у Вінницькому національному технічному університеті» 2013.

9. Вимоги щодо технічного захисту інформації в МКР з обмеженим доступом

Відсутні.

ДОДАТОК Б

Група класів «інформація»

```
public class InventoryItemViewModel
   {
      public Image Image { get; set; }
      public ProductInfo ProductInfo { get; set; }
      public AvailableStock AvailableStock { get; set; }
      public Purchasing Purchasing { get; set; }
      public PurchaseHistory PurchaseHistory { get; set; }
      public SalesStatistics Velocity { get; set; }
      public Profitabillity Profitabillity { get; set; } = new ProfitabillityInventoryItem();
      public MarketStats MarketStats { get; set; } = new MarketStatsInventoryItem();
        }
public class Image
   {
      public string Link
      {
         get
         {
            if (ImageLink != null)
               return ImageLink.Replace("_SL75_", "_SL150_");
            return ImageLink;
         }
      }
      public string BigImageLink
      {
         get
         {
            if (ImageLink != null)
```

```
                return ImageLink.Replace("_SL75_", "_SL300_");
            return ImageLink;
        }
    }

    public string ImageLink { get; set; }
  }
public class ProductInfo
  {
    public string ASIN { get; set; }
    public string Condition { get; set; }
    public string FNSKU { get; set; }
    public string SKU { get; set; }
    public string ProductName { get; set; }
    public string MfnAfn { get; set; }
    public string ProductSize { get; set; }
    public decimal PerUnitVolume { get; set; }
    public IEnumerable<KitsSkuViewModel> Kits { get; set; }
    public bool IsOversize { get; set; }
    public bool IsChild { get; set; }
    public string ParentAsin { get; set; }
    public string VendorName { get; set; }
    public decimal YourPrice { get; set; }
    public WeightDimension WeightDimension { get; set; }
    public IEnumerable<double> Dimensions { get; set; }
    public bool IsChangeHistory { get; set; }
    public decimal? AmazonReferralFee { get; set; }
    public decimal? FulfillmentFees { get; set; }
    public string Hash { get; set; }
  }
```

```
public class AvailableStock
    {
        public int AvailableMfn { get; internal set; }
        public int AvailableFba { get; internal set; }
        public int InboundQty { get; internal set; }
        public int ReservedFcTransfer { get; internal set; }
        public int ReservedFcProcessing { get; internal set; }
        public int TotalInventory { get; set; }
        public int MerchantWarehouseQty { get; set; }
    }
public class Purchasing
    {
        public decimal LastAverageCost { get; set; }
        public decimal LastCost { get; set; }

        private decimal _volCost;
        public decimal LastVolumeCost
        {
            get
            {
                if (_volCost > 0)
                    return _volCost + this.LastInternalLogisticsCost;
                return 0;
            }
            set
            {
                _volCost = value;
            }
        }
        public decimal VolumeCost => _volCost;
```

```
    public decimal LastInternalLogisticsCost { get; set; }
    public decimal LastUnitPrice { get; set; }
    public decimal OldLastCost => LastCost;
}
```

ДОДАТОК В

Лістинг контролера Inventory

```
public async Task<ItemsResponse> GetItems()
{
    var response = await this.searchStrategy.Get();
    var items = response.Items;
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    try
    {
        var asins = items.Select(i => i.Asin).Distinct();
        var skus = items.Select(s => s.Sku).Distinct();

        if (skus.Any() && asins.Any() && items.Any())
        {
            Dictionary<string, InventoryCallGetMatchingProductForIdResponse>
getMatchingProductForIdCall = null;
            Dictionary<string, PurchasingCost> costs = null;
            Dictionary<string, List<KitsSkuViewModel>> kits = null;
            Dictionary<string, AdvertisingInventoryItem> advertisings = null;
            Dictionary<string, SkuSettingsInventoryItem> skusSettings = null;
            Dictionary<string, decimal> logisticsCosts = null;
            Dictionary<string, string> vendorNames = null;
            Dictionary<string, bool> changeHistories = null;
            string profitPercentage = null;

            List<Task> tasks = new List<Task>();
            tasks.Add(Task.Run(() => profitPercentage =
GetProfitPercentageMethod()));
```

```
                tasks.Add(Task.Run(() => getMatchingProductForIdCall =
GetMatchingProductForIdCall(new ConcurrentBag<string>(asins))));
                tasks.Add(Task.Factory.StartNew(() => costs =
GetCostsByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => kits =
GetKitsByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => advertisings =
GetAdvertisingDataByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => amazonRanks =
GetAmazonRanksByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => skusSettings =
GetUsersSkuSettingsByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => eligibleOffers =
GetEligibleOffersByListAsins(items.Where(i => i.AfnFulfillableQuantity > 0).Select(i
=> i.Asin).Distinct().ToList())));
                tasks.Add(Task.Factory.StartNew(() => skusSalesVelocities =
GetSalesVelocitiesByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => logisticsCosts =
GetLastLogisticCostByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => vendorNames =
GetVendorNamesByListSkus(skus.ToList())));
                tasks.Add(Task.Factory.StartNew(() => changeHistories =
GetHistoricalItemsChangesByListSkus(skus.ToList())));
                Task.WaitAll(tasks.ToArray());
                tasks.Clear();
                Stopwatch stopwatch1 = new Stopwatch();
                stopwatch1.Start();
                try
                {
                   foreach (var sku in skus)
```

```
                {
                    var item = items.First(s => s.Sku == sku);
                    tasks.Add(Task.Factory.StartNew(() =>
                    {
                        List<Task> itemTasks = new List<Task>();
                        item.MarketStats = new MarketStatsInventoryItem();
                        item.Purchasing = new PurchasingInventoryItem();
                        item.PurchaseHistory = new PurchaseHistoryInventoryItem();
                        if (costs.ContainsKey(sku))
                        {
                            item.PurchaseHistory.Orders = costs[sku].Orders;
                            item.Purchasing.LastAverageCost =
costs[sku].LastAverageCost;
                            item.Purchasing.LastVolumeCost =
costs[sku].LastVolumeCost;
                            item.Purchasing.LastUnitPrice = costs[sku].LastUnitPrice;
                            item.Purchasing.LastCost = costs[sku].LastCost;
                        }
                        item.ProductInfo = new ProductInfoInventoryItem
                        {
                            ASIN = item.Asin,
                            FNSKU = item.Fnsku,
                            Condition = item.Condition,
                            MfnAfn = item.MfnListingExists == "Yes" ? "MFN" :
item.AfnListingExists == "Yes" ? "AFN" : null,
                            ProductName = item.ProductName,
                            ProductSize = item.ProductSizeTier,
                            SKU = item.Sku,
                            PerUnitVolume = item.PerUnitVolume ?? 0,
                            YourPrice = item.YourPrice ?? 0,
```

```
                    AmazonReferralFee = item.AmazonReferralFee ?? 0,
                    FulfillmentFees = item.FulfillmentFees ?? 0,
                    Hash = item.Sku.HashSHA1()
                };
                if (changeHistories.ContainsKey(sku))
                    item.ProductInfo.IsChangeHistory = changeHistories[sku];
                if (vendorNames.ContainsKey(sku))
                    item.ProductInfo.VendorName = vendorNames[sku];
                if (kits.ContainsKey(sku))
                    item.ProductInfo.Kits = kits[sku];
                else
                    item.ProductInfo.Kits = new List<KitsSkuViewModel>();
                if (advertisings.ContainsKey(sku))
                    item.Advertising = advertisings[sku];
                else
                    item.Advertising = new AdvertisingInventoryItem();
                item.MarketStats.Ranks = new
List<MarketStatsRankInventoryModel>();
                if (skusSettings.ContainsKey(sku))
                    item.Orders.Load(skusSettings[sku]);
                item.AvailableStock = new AvailableStockInventoryItem
                {
                    AvailableMfn = item.MfnFulfillableQuantity ?? 0,
                    AvailableFba = item.AfnFulfillableQuantity ?? 0,
                    InboundQty = item.InboundQty ?? 0,
                    ReservedFcProcessing = item.ReservedFcProcessing ?? 0,
                    ReservedFcTransfer = item.ReservedFcTransfers ?? 0,
                    TotalInventory = item.TotalInventory,
                    MerchantWarehouseQty = item.MerchantWarehouseQty ?? 0
                };
```

```
                    if (logisticsCosts.ContainsKey(sku))
                        item.Purchasing.LastInternalLogisticsCost =
logisticsCosts[sku];
                    if (item.ProductInfo.Kits != null && item.ProductInfo.Kits.Any())
                        item.Orders.LastCost = item.ProductInfo.Kits.Sum(s =>
s.Kit.UnitPrice);
                    else if (item.Purchasing.LastCost > 0)
                        item.Orders.LastCost = item.Purchasing.LastCost as decimal?;
                    if (itemTasks.Any())
                        Task.WaitAll(itemTasks.ToArray());
                }));
            }
            Task.WaitAll(tasks.ToArray());
        }
        finally
        {
            stopwatch1.Stop();
            Debug.WriteLine("MapProps elapsed: {0:ss\\.fff}",
stopwatch.Elapsed);
        }
        List<AmazonCategoriesKeyValue> amazonCategoriesKeyValues = null;
        var categoryIds = getMatchingProductForIdCall.Values.Where(s =>
s.SalesRanks != null).SelectMany(s => s.SalesRanks.Select(r =>
r.ProductCategoryId)).Distinct().ToList();
        using (var ctx = new AuroraServiceMiningContext())
            amazonCategoriesKeyValues =
ctx.AmazonCategoriesKeyValues.Where(i => categoryIds.Contains(i.Id)).ToList();
        foreach (var item in items)
        {
```

```
                EstimatedFee fee = new EstimatedFee { AmazonReferralFee =
item.AmazonReferralFee ?? 0, FulfillmentFees = item.FulfillmentFees ?? 0 };
                if (getMatchingProductForIdCall.ContainsKey(item.Asin))
                {
                    InventoryCallGetMatchingProductForIdResponse productForId =
getMatchingProductForIdCall[item.Asin];
                    try
                    {
                        item.Image = new ImageInventoryItem
                        {
                            ImageLink = productForId.ImageLink
                        };
                        item.ProductInfo.IsOversize = productForId.IsOversize;
                        item.ProductInfo.IsChild = productForId.IsChild;
                        item.ProductInfo.ParentAsin = productForId.ParentAsin;
                        item.ProductInfo.WeightDimension =
productForId.WeightDimension;
                        item.ProductInfo.Dimensions = productForId.Dimensions;
                        if (productForId.SalesRanks != null &&
productForId.SalesRanks.Any())
                            item.MarketStats.Ranks =
GenerateAmazonRanks(productForId.SalesRanks, amazonCategoriesKeyValues);
                        if (item.ProductInfo.ProductName == null)
                            item.ProductInfo.ProductName = productForId.ProductName;
                    }
                    catch (Exception ex)
                    {
                        Extensions.NotificationExtensions.AddLog(ex, "InventoryItems.cs
396 line " + JsonConvert.SerializeObject(productForId));
        }
```

```
                }
                CalculateProfitPercentage(item, profitPercentage);
            }
        }
        return true;
    }
    catch (Exception ex)
    {
        Extensions.NotificationExtensions.AddLog(ex,
$"LoadData({this.searchStrategy.SearchType}, {this.searchStrategy.Query},
{this.searchStrategy.OrderBy}, {this.searchStrategy.Skip},
{this.searchStrategy.Take})");
        return false;
    }
    finally
    {
        stopwatch.Stop();
        Debug.WriteLine("FillItemsNullData elapsed: {0:ss\\.fff}",
stopwatch.Elapsed);
    }
    return response;
}

public async Task<ActionResult>
RemoveFromAmazon(IEnumerable<RemoveItemModel> skus)
{
    try
    {
        var list = skus?.Distinct().Where(s => !string.IsNullOrEmpty(s)).ToList();
        if (list?.Any() ?? false)
```

```
    {
        var rnd = new Random();
        var job = new Job
        {
            JobTypeId = (int)JobTypes.RemoveItemFromAmazon,
            DateStarted = DateTime.UtcNow,
            TotalItems = skus.Count(),
        };
        _serviceMining.Entry(job).State = EntityState.Added;
        await _serviceMining.SaveChangesAsync();
        var feedQueueName = Constants.QueueFeedName;
        var message = new DeleteItemCommand
        {
            Skus = list,
            ContentType = Constants.FeedDelete,
            Environment = "pro",
            CompanyId = companyId,
            JobId = job.Id
        };
        var groupId = Convert.ToString(rnd.Next(int.MinValue, int.MaxValue));
        var jsonMessage = JsonConvert.SerializeObject(message);
        _queueService.SendMessage(jsonMessage, feedQueueName, groupId);
        return Json(new { Success = true });
    }
    else
    {
        return Json(new { Success = false, Message = "List empty." });
    }
}
catch (Exception ex)
```

```
        {
            return Json(new { Success = false, ex.Message });
        }
        }

        public async Task<JsonResult>
ChangeFulfillmentMethodForSku(ChangeFulfillmentMethod model)
        {
            try
            {
                var headers = new AmazonEnvelopeHeader
                {
                    MerchantIdentifier = feedWebService.LoginModel.MerchantId,
                    DocumentVersion = "1.01"
                };
                var results = await PostFeedProductWithoutResult(feedWebService, new
AmazonEnvelope
                {
                    MessageType = "Inventory",
                    Header = headers,
                    Message = new AmazonEnvelopeMessage[]
                    {
                    new AmazonEnvelopeMessage
                    {
                        MessageID = 1,
                        OperationType = "Update",
                        Inventory = new AmazonEnvelopeMessageInventory
                        {
                            SKU = sku,
                            SwitchFulfillmentTo = switchFulfillment
```

```
                }
            }
            },
        }, feedWebService.LoginModel,
"_POST_INVENTORY_AVAILABILITY_DATA_");
            return new { Success = true };
        }
        catch (Exception ex)
        {
            return new { Success = false, Message = ex.Message };
    }
    }
    public async Task<JsonResult> ChangePrice(ChangePrice model)
    {
        try
        {
            if (list.Any(l => l.PropName != "Min Price" && l.PropName != "Max
Price"))
                throw new Exception("Invalid key.");

            // find itemStatus
            // using EF cause SqlExtensions can't convert int32 to enum value
            var itemStatus =
serviceMiningContext.Database.SqlQuery<FbaInboundShipmentItemsStatus>($"SELE
CT * FROM FbaInboundShipmentItemsStatuses WHERE Sku =
'{sku}'").FirstOrDefault();
            if (itemStatus == null)
            {
                //create if doesn't exist
                itemStatus = new FbaInboundShipmentItemsStatus();
```

```
                itemStatus.Id = Guid.NewGuid();
                itemStatus.Sku = sku;
                // insert new row
                SqlExtensions.MySqlExecuteCommand($"INSERT INTO
FbaInboundShipmentItemsStatuses (Id, Sku) VALUES ('{itemStatus.Id}',
'{itemStatus.Sku}')", ConnectionsOptions.AuroraServiceMiningDbConnection);
            }
            if (list.Any())
            {
                var parts = new List<string>();
                foreach (var price in list)
                {
                    switch (price.PropName)
                    {
                        case "Min Price":
                            itemStatus.MinPrice = price.Price;
                            parts.Add($"MinPrice = {price.Price}");
                            break;
                        case "Max Price":
                            itemStatus.MaxPrice = price.Price;
                            parts.Add($"MaxPrice = {price.Price}");
                            break;
                    }
                }
                if (parts.Any() && parts.Count < 3)
                {
                    parts.Add("UpdateStatus = 1"); // make UpdateStatus = Sent
                    parts.Add($"UpdatedDate = '{DateTime.Now.ToString("yyyy-MM-dd
HH:mm")}'");
```

```
                var sqlCommand = $"UPDATE FbaInboundShipmentItemsStatuses
SET {string.Join(", ", parts)} WHERE Sku = '{sku}'";
                SqlExtensions.MySqlExecuteCommand(sqlCommand,
ConnectionsOptions.AuroraServiceMiningDbConnection);
                // TO DO: rewrite to queue-approach
                var priceNow = await GetSalesPrice90BySkuAsync(sku);
                string propName = SiteSettings.AppeagleXApiKey.ToString();
                var setting = await new
AuroraManagmentContext().SiteSettings.SingleOrDefaultAsync(w => w.PropName ==
propName);
                if (setting.PropValue != string.Empty)
                {
                   if (list?.Any() ?? false)
                   {
                      var messages = new List<MinMaxPriceMessage>();
                      foreach (var price in list)
                      {
                         var message = new MinMaxPriceMessage
                         {
                            Sku = sku,
                            PropName = price.PropName,
                            Value = price.Price,
                            CurrentPrice = priceNow.PriceNow,
                            MinPrice = itemStatus.MinPrice ?? 0,
                            MaxPrice = itemStatus.MaxPrice ?? 0
                         };
                         messages.Add(message);
                      }
                      var rnd = new Random();
```

```
                    var groupId = Convert.ToString(rnd.Next(int.MinValue,
int.MaxValue));
                    var jsonMessage = JsonConvert.SerializeObject(messages);
                    _queueService.SendMessage(jsonMessage, queueName, groupId);
                }
            }
            else
            {
                foreach (var price in list)
                {
                    var sellersnapStore = _sellersnap.GetStores();
                    ListingSellersnapRequest listing = null;
                    var listings = new List<ListingSellersnapRequest>();
                    if (price.PropName == "Min Price")
                    {
                        listing = new ListingSellersnapRequest
                        {
                            repricing = true,
                            sku = sku,
                            min_price = price.Price
                        };
                    }
                    else if (price.PropName == "Max Price")
    {
                        listing = new ListingSellersnapRequest
                        {
                            repricing = true,
                            sku = sku,
                            max_price = price.Price
                        };
```

```
                    }
                    listings.Add(listing);
                    _sellersnap.UpdatingListings(listings,
sellersnapStore.data[0].store_id);
                }
            }
        }
        return new { Success = true };
    }
    catch (Exception ex)
    {
        _logger.Fatal($"{ex.Message} ({ex.StackTrace})");
        return new { Success = false, Message = $"Error is occured during add
{sku} to updating price queue. ({ex.Message})" };
    }
}
public async Task<JsonResult> UpdateCaseQty(stringsku, intqty = 0)
{
    try
    {
        using (var context = serviceMiningReadOnlyContext)
{
            var data = await
context.FbaInboundShipmentItemsStatuses.FirstOrDefaultAsync(i => i.Sku == sku);
            if (data != null)
            {
                data.Case = qty;
                context.Entry(data).State = EntityState.Modified;
            }
```

```
                else
                {
                    Sku sku1 = await context.Skus.FirstOrDefaultAsync(s => s.Sku1 ==
sku);
                    context.FbaInboundShipmentItemsStatuses.Add(new
FbaInboundShipmentItemsStatus
                    {
                        Sku = sku,
                        Case = qty,
                        Id = Guid.NewGuid(),
                        Status = 0,
                        Sku1 = sku1
                    });
                }
                await context.SaveChangesAsync();
                return new { Success = true };
            }
        }
        catch (Exception ex)
        {
            return new { Success = false, ex.Message };
    }
    }
```

ДОДАТОК Г

Лістинг контролера PrepareFbaShipment

```
public async Task<ActionResult> AddToFBA(List<AddToFbaModel> list)
{
    try
    {
        var file = list.Select(w => new
        {
            Sku = w.SKU,
            Qty = w.Quantity
        }).ToList();
        var fileName = (DateTime.Now.ToString() +
_companyId.ToString()).HashSHA1();
        if (file.Count > 0)
        {
            var content = file.Distinct().Aggregate($"Sku\tQty\n", (a, b) => { return a
= a + $"{b.Sku}\t{b.Qty}\n"; });
            var path = $"FbaShipPrep/{_companyId}/{fileName}.txt";
            var ms = new System.IO.MemoryStream();
            var ret = content.SerializeToStream(ms);
            await _storageService.SaveObjectAsync(path, ms);
        }
        return new
        {
            Success = true,
            Message = fileName
        };
    }
    catch (Exception ex)
```

```
            {
                return new
                {
        Success = false,
                    ex.Message
                };
            }
        }

        public async Task<ActionResult>
GetInboundGuidanceAndPrepInstructionsForSKU(ShipmentSetting Setting)
        {
            try
            {
BoxHub.UpdateGetInboundGuidanceAndPrepInstructionsForSKUProgress(progressCo
nnectionId, 1);
                SellerSKUList SellerSKULists = new SellerSKUList();
                Setting.ShipmentItems = new List<ShipmentItem>();
                foreach (var item in Items)
                {
                    var i = Setting.ShipmentItems.SingleOrDefault(w => w.SKU ==
item.SKU);
                    if (i != null)
                    {
                        i.Quantity += item.Quantity;
                        continue;
                    }
                    Setting.ShipmentItems.Add(item);
                }
                SellerSKULists.Id.AddRange(Setting.ShipmentItems.Select(w => w.SKU));
```

```
            foreach (var item in Setting.ShipmentItems)
            {
                if (item.CustomQuantityInCase != 0)
                {
                    item.QuantityInCase = item.CustomQuantityInCase;
                }
                else
                {
                    var caseQty =
_sellermogulContext.FbaInboundShipmentItemsStatuses.Where(w => w.Sku ==
item.SKU).SingleOrDefault()?.Case;
                    if (caseQty != null)
                        item.QuantityInCase = caseQty == 0 ? 1 : (decimal)caseQty;
                    else
                        item.QuantityInCase = 1;
                }
            }
            var address = await _sellermogulContext.FbaAddresses.Where(w =>
w.CompanyId == UserInfo.CompanyId.Value).FirstOrDefaultAsync();
BoxHub.UpdateGetInboundGuidanceAndPrepInstructionsForSKUProgress(progressCo
nnectionId, 10);
            foreach (var items in SellerSKULists.Id.Partition(50))
            {
                SellerSKUList SellerSKUListsPartition = new SellerSKUList();
                var cacheInboundGuidanceCollections = new
List<InboundGuidanceItem>();
                foreach (var item in items)
                {
                    var cacheItem = _memoryCacheService.Get($"inbound-guidance-
{item}");
```

```
                if (cacheItem != null)
                {

cacheInboundGuidanceCollections.Add((InboundGuidanceItem)cacheItem);
                }
            }
            var skusNeedInboundGuidance = items.Where(w =>
!cacheInboundGuidanceCollections.Any(s => s.SellerSKU == w)).ToList();
            SellerSKUListsPartition.Id.AddRange(skusNeedInboundGuidance);
            var responceInboundGuidance = new
GetInboundGuidanceForSKUResult();
            var InboundGuidanceCollections = new List<InboundGuidanceItem>();
            if (skusNeedInboundGuidance.Count > 0)
            {
                responceInboundGuidance = await
ExponentialBackoffFunction.AsyncOperation(() =>
ShipmentExtensions.GetInboundGuidanceForSKU(MwsLoginModel,
SellerSKUListsPartition, UserInfo.CompanyId.Value), 1000);
                InboundGuidanceCollections =
responceInboundGuidance.SKUInboundGuidanceList.IsSetSKUInboundGuidance() ?
responceInboundGuidance.SKUInboundGuidanceList
                    .SKUInboundGuidance.Select(w => new InboundGuidanceItem
                    {
                        SellerSKU = w.SellerSKU,
                        ASIN = w.ASIN,
                        InboundGuidance = w.InboundGuidance,
                        GuidanceReasonList =
w.GuidanceReasonList.GuidanceReason.Aggregate("", (a, b) => { return a = a + b; })
                    }).ToList() : null;
            }
```

```
            foreach (var item in InboundGuidanceCollections)
                _memoryCacheService.Store($"inbound-guidance-{item.SellerSKU}",
item, DateTime.Now.AddHours(1));
            if (cacheInboundGuidanceCollections.Count > 0)
InboundGuidanceCollections.AddRange(cacheInboundGuidanceCollections);
            //prep instractions
            var cachePrepInstructionCollections = new List<SKUPrepInstructions>();
            var PrepInstructionCollections = new List<PrepInstructionItem>();
            foreach (var item in items)
            {
                var cacheItem = _memoryCacheService.Get($"sku-prep-instractions-
{item}");
                if (cacheItem != null)
                {

cachePrepInstructionCollections.Add((SKUPrepInstructions)cacheItem);
                }
            }
            var skusNeedSKUPrepInstructions = items.Where(w =>
!cachePrepInstructionCollections.Any(s => s.SellerSKU == w)).ToList();
PrepInstructionCollections.AddRange(cachePrepInstructionCollections.Select(s => new
PrepInstructionItem
            {
                SellerSKU = s.SellerSKU,
                ASIN = s.ASIN,
                BarcodeInstruction = s.BarcodeInstruction,
                PrepGuidance = s.PrepGuidance,
                PrepInstruction = s.PrepInstructionList.PrepInstruction.Aggregate("",
(a, b) => a = a + (a == "" ? b : $",{b}")),
                PrepInstructionList = s.PrepInstructionList
```

```
            }));

SKUPrepInstructionsCollections.AddRange(cachePrepInstructionCollections);
            if (skusNeedSKUPrepInstructions.Count > 0)
            {
                var SellerSKUListsByPrepInstructionPartition = new SellerSKUList();

SellerSKUListsByPrepInstructionPartition.Id.AddRange(skusNeedSKUPrepInstruction
s);
                var responcePrepInstructions = await await
ExponentialBackoffFunction.AsyncOperation(() =>
ShipmentExtensions.GetPrepInstructionsForSKU(MwsLoginModel,
SellerSKUListsByPrepInstructionPartition,UserInfo.CompanyId.Value,
address.ShipToCountryCode), 1000);
                if (responcePrepInstructions.IsSetSKUPrepInstructionsList())
                {
                    foreach (var item in
responcePrepInstructions.SKUPrepInstructionsList.SKUPrepInstructions)
                    {
                        _memoryCacheService.Store($"sku-prep-instractions-
{item.SellerSKU}", item, DateTime.Now.AddHours(1));
                    }
                }
                var PrepInstructions =
responcePrepInstructions.IsSetSKUPrepInstructionsList() ? responcePrepInstructions.
                    SKUPrepInstructionsList.SKUPrepInstructions.Select(w => new
PrepInstructionItem
                    {
                        SellerSKU = w.SellerSKU,
                        ASIN = w.ASIN,
```

```
                    BarcodeInstruction = w.BarcodeInstruction,
                    PrepGuidance = w.PrepGuidance,
                    PrepInstruction =
w.PrepInstructionList.PrepInstruction.Aggregate("", (a, b) => a = a + (a == "" ? b :
$",{b}")),
                    PrepInstructionList = w.PrepInstructionList
                }).ToList() : null;
            PrepInstructionCollections.AddRange(PrepInstructions);
SKUPrepInstructionsCollections.AddRange(responcePrepInstructions.SKUPrepInstruct
ionsList.SKUPrepInstructions);
InvalidSKUCollections.AddRange(responcePrepInstructions.InvalidSKUList.InvalidSK
U);
        }
        var query = (from i in InboundGuidanceCollections
                     join p in PrepInstructionCollections
                     on i.SellerSKU equals p.SellerSKU
                     select new InboundGuidanceAndPrepInstructionsModel
                     {
                        SellerSKU = i.SellerSKU,
                        ASIN = i.ASIN,
                        InboundGuidance = i.InboundGuidance,
                        GuidanceReasonList = i.GuidanceReasonList,
                        BarcodeInstruction = p.BarcodeInstruction,
                        PrepGuidance = p.PrepGuidance,
                        PrepInstructionList = p.PrepInstruction.Split(','),
                        InstructionList = p.PrepInstructionList,
                        CaseQty = Setting.ShipmentItems.Where(w => w.SKU ==
i.SellerSKU).SingleOrDefault().QuantityInCase,
                        Qty = Setting.ShipmentItems.Where(w => w.SKU ==
i.SellerSKU).SingleOrDefault().Quantity,
```

```
                                Oversize = false,
                                FrontLoad = 0,
                                BackLoad = 0,
                                MerchQty = 0,
                                Action = Constants.None,
                                Title =
_sellermogulContext.GetFbaMyiAllInventoryData.SingleOrDefault(w => w.Sku ==
i.SellerSKU)?.ProductName ?? string.Empty,
                                TempCaseQty = false,
                                BillId = Setting.SkusToFba?.Where(w => w.Sku ==
i.SellerSKU).FirstOrDefault()?.BillId,
                                DocNumber = Setting.SkusToFba?.Where(w => w.Sku ==
i.SellerSKU).FirstOrDefault()?.DocNumber
                            }).ToList();
                InboundGuidanceAndPrepInstructionsCollections.AddRange(query);
            }
BoxHub.UpdateGetInboundGuidanceAndPrepInstructionsForSKUProgress(progressCo
nnectionId, 40);
            foreach (var items in
InboundGuidanceAndPrepInstructionsCollections.Partition(5))
            {
                var GetMatchingProductForIdResultCollections = new
List<GetMatchingProductForIdResult>();
                foreach (var item in items)
                {
                    var cacheItem =
_memoryCacheService.Get($"GetMatchingProductForIdResult-{item.SellerSKU}");
                    if (cacheItem != null)
                    {
GetMatchingProductForIdResultCollections.Add((GetMatchingProductForIdResult)cac
```

```
heItem);
                }
            }

            var needSkus = items.Where(w =>
!GetMatchingProductForIdResultCollections.Any(x => x.Id == w.SellerSKU)).Select(s
=> s.SellerSKU).ToList();
            if (needSkus.Count > 0)
            {
                await Task.Delay(1100);
                var responce =
MwsProductsClientExtended.GetMatchingProductForId(needSkus,
IdProductType.SellerSKU);
                if (responce.IsSetGetMatchingProductForIdResult())
                {
GetMatchingProductForIdResultCollections.AddRange(responce.GetMatchingProductF
orIdResult);
                }
            }
            GetMatchingProductForIdResultCollections.ForEach(x =>
            {
                if (x.IsSetProducts())
                {
                    var cacheItem =
_memoryCacheService.Get($"GetMatchingProductForIdResult-{x.Id}");
                    if (cacheItem == null)
                        _memoryCacheService.Store($"GetMatchingProductForIdResult-
{x.Id}", x, DateTime.Now.AddHours(2));
                    XmlElement xml = x.Products.Product[0].AttributeSets.Any[0] as
XmlElement;
```

```
                    XmlElement packageDimensionNode =
xml.GetElementsByTagName("ns2:PackageDimensions").Item(0) as XmlElement;
                    List<double> dimensionList = new List<double> { };
                    if (packageDimensionNode != null)
                    {
dimensionList.Add(Convert.ToDouble(packageDimensionNode.GetElementsByTagNa
me("ns2:Height").Item(0)?.InnerText ?? "0"));
dimensionList.Add(Convert.ToDouble(packageDimensionNode.GetElementsByTagNa
me("ns2:Width").Item(0)?.InnerText ??
"0"));dimensionList.Add(Convert.ToDouble(packageDimensionNode.GetElementsByT
agName("ns2:Length").Item(0)?.InnerText ?? "0"));
                        dimensionList = dimensionList.OrderByDescending(val =>
val).ToList();
                        if (dimensionList[0] > 18 || dimensionList[1] > 14 ||
dimensionList[2] > 8)
                        {
                            var item = items.SingleOrDefault(w => w.SellerSKU == x.Id);
                            item.Oversize = true;
                        }
                    }
                }
            });
        }
BoxHub.UpdateGetInboundGuidanceAndPrepInstructionsForSKUProgress(progressCo
nnectionId, 80);
        var FrontLoadVelocity = await
_managmentContext.SiteSettings.SingleOrDefaultAsync(w => w.PropName ==
"FrontLoadVelocity");
        var FrontLoadDays = await
_managmentContext.SiteSettings.SingleOrDefaultAsync(w => w.PropName ==
```

```
"FrontLoadDays");
            var FBAStockDays = await
_managmentContext.SiteSettings.SingleOrDefaultAsync(w => w.PropName ==
"FBAStockDays");
            var Velocity = new Dictionary<string, dynamic>();
            var AfnFulfillableQuantitys = new Dictionary<string, dynamic>();
            InboundGuidanceAndPrepInstructionsCollections.ForEach(x =>
            {
               // get velocity
               var cacheVelocity = _memoryCacheService.Get($"velocity-
{x.SellerSKU}");
               if (cacheVelocity == null)
               {
                  var velocity = GetVelocity(x.SellerSKU);
                  _memoryCacheService.Store($"velocity-{x.SellerSKU}", velocity,
DateTime.Now.AddHours(3));
                  Velocity.Add(x.SellerSKU, velocity);
               }
               else
                  Velocity.Add(x.SellerSKU, (Dictionary<string,
decimal>)cacheVelocity);
               // get AfnFulfillableQuantitys
               var cacheGetAfnFulfillableQuantityBySku =
_memoryCacheService.Get($"GetAfnFulfillableQuantityBySku-{x.SellerSKU}");
               if (cacheGetAfnFulfillableQuantityBySku == null)
               {
                  var afnFulfillableQty =
GetAfnFulfillableQuantityBySku(x.SellerSKU);
                  AfnFulfillableQuantitys.Add(x.SellerSKU, afnFulfillableQty);
                  _memoryCacheService.Store($"GetAfnFulfillableQuantityBySku-
```

```
{x.SellerSKU}", afnFulfillableQty, DateTime.Now.AddHours(3));
                }
                else
                    AfnFulfillableQuantitys.Add(x.SellerSKU,
(int?)cacheGetAfnFulfillableQuantityBySku);
            });
        BoxHub.UpdateGetInboundGuidanceAndPrepInstructionsForSKUProgress(progr
essConnectionId, 95);
            InboundGuidanceAndPrepInstructionsCollections.ForEach(x =>
            {
                x.StatusCollection.Add(PrepInstructionsModelStatus.NONE);
                x.Status = PrepInstructionsModelStatus.NONE;
                if (Setting.MerchantWarehouse == "NO" && Setting.QuantityInCase ==
"YES")
                {
                    if (x.Qty % x.CaseQty != 0)
                    {
                        x.TempCaseQty = true;
                        x.CaseQty = 1;
                    }
                }
                if (x.Oversize)
                {
                    x.StatusCollection.Add(PrepInstructionsModelStatus.OVERSIZE);
                    x.Status = PrepInstructionsModelStatus.OVERSIZE;
                }
                if (x.InboundGuidance != Constants.InboundOk)
                {

x.StatusCollection.Add(PrepInstructionsModelStatus.INBOUND_ERROR);
```

```
                    x.Status = PrepInstructionsModelStatus.INBOUND_ERROR;
                }
                x.Status = x.StatusCollection.OrderBy(s => s).FirstOrDefault();
                x.StatusCollection = x.StatusCollection.OrderBy(s => s).ToList();
            });
        BoxHub.UpdateGetInboundGuidanceAndPrepInstructionsForSKUProgress(progr
essConnectionId, 100);
            return Json(new
            {
                Success = true,
                Message = InboundGuidanceAndPrepInstructionsCollections.OrderBy(w
=> w.Status).ThenByDescending(w => w.InboundGuidance),
                Model = SKUPrepInstructionsCollections,
                InvalidSKU = InvalidSKUCollections,
                Velocity = Velocity,
                FrontLoadDays = FrontLoadDays.PropValue,
                FBAStockDays = FBAStockDays.PropValue,
                AfnFulfillableQuantitys = AfnFulfillableQuantitys
            }, JsonRequestBehavior.AllowGet);
        }
        catch (Exception ex)
        {
            _loggerService.Info($"|GetInboundGuidanceAndPrepInstructionsForSKU|
<br> {ex.Message} ");
            return Json(new
            {
                Success = false,
                Message = ex.Message
            }, JsonRequestBehavior.AllowGet);
        }
```

```
        }
        public async Task<ActionResult>
CreateInboundShipmentPlan(InboundShipmentPlanplan)
        {
           try
           {
              var InboundGuidanceAndPrepInstructionsByMerch = new
List<InboundGuidanceAndPrepInstructionsModel>();
              var InboundGuidanceAndPrepInstructionsByFBAQty = new
List<InboundGuidanceAndPrepInstructionsModel>();
              Dictionary<CaseLoadType, List<InboundShipmentPlan>>
plansByLoadType = new Dictionary<CaseLoadType, List<InboundShipmentPlan>>();
              Guid newDbCollectionId = Guid.Empty;
              InboundGuidanceAndPrepInstructionsByMerch =
InboundGuidanceAndPrepInstructionsResponse.Where(w => w.MerchQty > 0).Select(s
=> new InboundGuidanceAndPrepInstructionsModel
              {
                 Action = s.Action,
                 ASIN = s.ASIN,
                 BarcodeInstruction = s.BarcodeInstruction,
                 CaseQty = s.CaseQty,
                 GuidanceReasonList = s.GuidanceReasonList,
                 InboundGuidance = s.InboundGuidance,
                 InstructionList = s.InstructionList,
                 Oversize = s.Oversize,
                 PrepGuidance = s.PrepGuidance,
                 PrepInstructionList = s.PrepInstructionList,
                 SellerSKU = s.SellerSKU,
                 Status = s.Status,
                 StatusCollection = s.StatusCollection,
```

```
                Title = s.Title,
                Qty = s.MerchQty
            }).ToList();
            InboundGuidanceAndPrepInstructionsByFBAQty =
InboundGuidanceAndPrepInstructionsResponse.Where(w => w.FBAQty > 0).Select(s
=> new InboundGuidanceAndPrepInstructionsModel
            {
                Action = s.Action,
                ASIN = s.ASIN,
                BarcodeInstruction = s.BarcodeInstruction,
                CaseQty = s.CaseQty,
                GuidanceReasonList = s.GuidanceReasonList,
                InboundGuidance = s.InboundGuidance,
                InstructionList = s.InstructionList,
                Oversize = s.Oversize,
                PrepGuidance = s.PrepGuidance,
                PrepInstructionList = s.PrepInstructionList,
                SellerSKU = s.SellerSKU,
                Status = s.Status,
                StatusCollection = s.StatusCollection,
                Title = s.Title,
                Qty = s.FBAQty
            }).ToList();
            if (Setting.MerchantWarehouse == Constants.Yes)
            {
                var plansByEmpty = await CreateInboundShipmentPlan(Setting,
InboundGuidanceAndPrepInstructionsByFBAQty, skuPrepInstructions,
CaseLoadType.Empty, Redistributions);
                plansByLoadType.Add(CaseLoadType.Empty, plansByEmpty);
            }
```

```
            else if (Setting.MerchantWarehouse == Constants.No)
            {
                var plansByEmpty = await CreateInboundShipmentPlan(Setting,
InboundGuidanceAndPrepInstructionsResponse, skuPrepInstructions,
CaseLoadType.Empty, Redistributions);
                plansByLoadType.Add(CaseLoadType.Empty, plansByEmpty);
            }
            List<ShipmentItem> SkusQty = new List<ShipmentItem>();
            List<dynamic> plans = new List<dynamic>();
            if (Setting.ShipmentItems.Where(w => w.Error != null).Count() == 0)
            {
                foreach (var planByKey in plansByLoadType)
                {
                    foreach (var item in planByKey.Value)
                    {
                        var dic = item.Items.member.Select(x => new ShipmentItem
                        {
                            SKU = x.SellerSKU,
                            Quantity = x.Quantity
                        });
                        foreach (var d in dic)
                        {
                            var i = SkusQty.SingleOrDefault(w => w.SKU == d.SKU);
                            if (i == null)
                                SkusQty.Add(d);
                            else
                            {
                                i.Quantity += d.Quantity;
                            }
                        }
```

```
                    plans.AddRange(item.Items.member.Select(w => new
                    {
                        ShipmentId = item.ShipmentId,
                        DestinationFulfillmentCenterId =
item.DestinationFulfillmentCenterId,
                        LabelPrepType = item.LabelPrepType,
                        Sku = w.SellerSKU,
                        Quantity = w.Quantity
                    }));
                }
            }
        }
        bool fix = false;
        if (Setting.AutoFix)
        {
            var FixedItems = await FixError(Setting, progressConnectionId);
            if (FixedItems.Count > 0)
            {
                foreach (var item in FixedItems)
                {
                    var sItem = Setting.ShipmentItems.SingleOrDefault(w => w.SKU ==
item.Sku);
                    if (sItem != null)
                    {
                        sItem.SKU = item.NewSku;
                    }
                }
                fix = true;
            }
        }
```

```
            var missItems = Setting.ShipmentItems.Where(s => !SkusQty.Any(x =>
x.SKU == s.SKU)).ToList();
            return Json(new
            {
               Success = true,
               Message = newDbCollectionId,
               ErrorItems = Setting.ShipmentItems.Where(w => w.Error !=
null).Select(s => new { Sku = s.SKU, Error = s.Error }).OrderBy(s => s.Error),
               Plans = plans,
               plansByLoadType = plansByLoadType.ToList(),
               Setting = Setting,
               skuPrepInstructions = skuPrepInstructions,
               InboundGuidanceAndPrepInstructionsByMerch =
InboundGuidanceAndPrepInstructionsByMerch,
               Fix = fix,
               missItems
            }, JsonRequestBehavior.AllowGet);
         }
         catch (Exception ex)
         {
            return Json(new
            {
               Success = false,
               Message = ex.Message
            }, JsonRequestBehavior.AllowGet); }
```

ДОДАТОК Д

Лістинг контроллера Inbound FBA Shipments

```
public async Task<ActionResult> GetAllShipmentCollections(int page, int pageSize, string nameFilter = null, string skuFilter = null)
{
    try
    {
        _sellermogulContext.Configuration.ProxyCreationEnabled = false;
        var allDbFbaInboundShipmentCollections = _sellermogulContext.FbaShipmentCollections.Where(c => c.CompanyId == UserInfo.CompanyId);
        if (!String.IsNullOrEmpty(nameFilter))
        {
            allDbFbaInboundShipmentCollections = allDbFbaInboundShipmentCollections.Where(c => c.Name.Contains(nameFilter));
        }
        if (!String.IsNullOrEmpty(skuFilter))
        {
            allDbFbaInboundShipmentCollections = allDbFbaInboundShipmentCollections.Where(c => _sellermogulContext.FbaInboundShipments.Any(s => s.FbaShipmentCollectionId == c.Id && _sellermogulContext.FbaInboundShipmentItems.Any(i => i.FbaInboundShipmentId == s.Id && i.Sku.Contains(skuFilter.ToUpper()))));
        }
        allDbFbaInboundShipmentCollections = allDbFbaInboundShipmentCollections.OrderByDescending(c => c.Date);
        var dbFbaInboundShipmentCollections = await allDbFbaInboundShipmentCollections.
            Skip((page - 1) * pageSize).
```

```
            Take(pageSize).
            ToListAsync();
        var collectionsIds = dbFbaInboundShipmentCollections.Select(c => c.Id);
        var shipments = await _sellermogulContext.FbaInboundShipments.Where(s
=> collectionsIds.Contains(s.FbaShipmentCollectionId)).Include(i =>
i.FbaInboundShipmentItems).ToListAsync();
        var shipmentIds = shipments.Select(c => c.Id);
        var parcels = await _sellermogulContext.FbaInboundParcels.Where(s =>
shipmentIds.Contains(s.ShipmentId)).Include(i =>
i.FbaInboundParcelItems).ToListAsync();
        var fbaInboundShipmentCollections =
dbFbaInboundShipmentCollections.Select(c =>
            {
                var totalScanned = parcels.Where(s => s.CollectionId == c.Id).Sum(s
=> s.FbaInboundParcelItems.Sum(si => si.Scanned));
                var totalShipmentItemsQTY = shipments.Where(s =>
s.FbaShipmentCollectionId == c.Id).Sum(s => s.FbaInboundShipmentItems.Sum(si =>
si.Qty));
                //Amazon update status
                var amazonUpdStatus = 0;    //0: Updated; 1: partial updated; 2:
unupdated
                if (shipments.Where(s => s.FbaShipmentCollectionId ==
c.Id).SelectMany(s => s.FbaInboundShipmentItems).Any(si => si.AmazonUpdateStatus
== (int)AmazonUpdateStatuses.NotUpdated))
                {
                    if (shipments.Where(s => s.FbaShipmentCollectionId ==
c.Id).SelectMany(s => s.FbaInboundShipmentItems).Any(si => si.AmazonUpdateStatus
== (int)AmazonUpdateStatuses.Updated))
                    {
                        amazonUpdStatus = 1;
```

```
                    }
                    else
                    {
                        amazonUpdStatus = 2;
                    }
                }
                //var totalUpdatedItems = shipments.Where(s =>
s.FbaShipmentCollectionId == c.Id).SelectMany(s =>
s.FbaInboundShipmentItems).Where(si => (si.AmazonUpdateStatus ?? 1) ==
(int)AmazonUpdateStatuses.Updated).Count();
                int totalSkuItemsQty = 0;
                decimal totalSkuItemsScanned = 0;
                if (!String.IsNullOrEmpty(skuFilter))
                {
                    totalSkuItemsQty = shipments.Where(s =>
s.FbaShipmentCollectionId == c.Id).Sum(s => s.FbaInboundShipmentItems.Where(si
=> si.Sku.Contains(skuFilter.ToUpper())).Sum(si => si.Qty));
                    totalSkuItemsScanned = parcels.Where(s => s.CollectionId ==
c.Id).Sum(s => s.FbaInboundParcelItems.Where(pi =>
pi.Sku.Contains(skuFilter.ToUpper())).Sum(si => si.Scanned));
                }
                return new
                {
                    Id = c.Id,
                    Name = c.Name,
                    CreatedDate = c.Date,
                    TotalShipments = c.FbaInboundShipments?.Count ?? 0,
                    TotalShipmentItems = c.FbaInboundShipments?.Sum(s =>
s.FbaInboundShipmentItems.Count) ?? 0,
```

```
                    TotalTransportAmount = c.FbaInboundShipments?.Sum(s =>
s.TransportAmmount ?? 0) ?? 0,
                    AmazonUpdStatus = amazonUpdStatus,
                    TotalShipmentQty = totalShipmentItemsQTY,
                    Completed = String.Format("{0:0}", totalScanned /
(totalShipmentItemsQTY == 0 ? 1 : totalShipmentItemsQTY) * 100),
                    TotalSkuQty = totalSkuItemsQty,
                    TotalSkuScanned = totalSkuItemsScanned,
                    c.DeliverrPlanId
                };
            });
        return Json(new
        {
            Success = true,
            TotalData = allDbFbaInboundShipmentCollections.Count(),
            Message = fbaInboundShipmentCollections
        }, JsonRequestBehavior.AllowGet);
    }
    catch (Exception exception)
    {
        WriteExceptionToLog("GetAllShipmentCollections", null, exception);
        return Json(new { Success = false, Message = exception.Message },
JsonRequestBehavior.AllowGet); } }

    public async Task<ActionResult> GetShipmentCollectionData(Guid collectionId)
    {
        try
        {
            _sellermogulContext.Configuration.ProxyCreationEnabled = false;
            var aws = new AWSService(AwsLoginModel);
```

```
            var listOfData = new List<object>();
            var asinUpcs = new List<AsinUpcsModel>();
            var amazonResults = new List<GetMatchingProductForIdResult>();
            Dictionary<string, string> asinImage = new Dictionary<string, string>();
            List<string> skuMissingDimension = new List<string>();
            var shipFrom = await _sellermogulContext.FbaAddresses.FirstAsync();
            var fbaCollection = await
_sellermogulContext.FbaShipmentCollections.SingleAsync(c => c.Id == collectionId);
            var fbaInboundShipments =
                await _sellermogulContext.FbaInboundShipments.Where(s =>
s.FbaShipmentCollectionId == collectionId).Include(s =>
s.FbaInboundShipmentItems).ToListAsync();
            var fbaInboundMerchant = await
GetFbaIboundMerchantsByCollection(collectionId);
            if (!fbaInboundShipments.Any()) return Json(new { Success = false,
Message = "Shipment Is empty" }, JsonRequestBehavior.AllowGet);
            var skuItemStatuses = await
_sellermogulContext.FbaInboundShipmentItemsStatuses.ToListAsync();
            var skus =
                fbaInboundShipments.
                SelectMany(s => s.FbaInboundShipmentItems.Select(si => si.Sku)).
                Concat(fbaInboundMerchant.FbaInboundShipmentItems.Select(si =>
si.Sku)).
                Distinct().
                ToList();
            var asins = fbaInboundShipments.SelectMany(s =>
s.FbaInboundShipmentItems.Select(si => si.Asin)).Distinct().Take(200).ToList();
            var allKits = await new
KitsExtension(_sellermogulContext).GetKitsBySkusAsync(skus);
            if (fbaInboundMerchant.FbaInboundShipmentItems.Count > 0)
```

```
                fbaInboundShipments.Add(fbaInboundMerchant);
            foreach (var fbaInboundShipment in fbaInboundShipments)
            {
                FbaInboundShipment internalFbaInbounShipment =
fbaInboundShipment;
                var shipmentListViewModel = new
List<FbaInboundShipmentItemViewModel>();
                var shipmentItems =
internalFbaInbounShipment.FbaInboundShipmentItems;
                var shipmentSkus = shipmentItems.Select(s => s.Sku).ToList();
                var missingSkus = shipmentItems.Where(i =>
String.IsNullOrEmpty(i.Title)).Select(i => i.Sku).ToList();
                var missingSkuTitles = await
_sellermogulContext.GetFbaMyiAllInventoryData.Where(m =>
missingSkus.Contains(m.Sku)).ToListAsync();
                foreach (var item in shipmentItems)
                {
                    var viewModel = new FbaInboundShipmentItemViewModel();
                    viewModel.CopyFrom(item);
                    viewModel.Kits = allKits.Where(k => k.Sku == item.Sku).ToList();
                    var statusInfo = skuItemStatuses.FirstOrDefault(s => s.Sku ==
item.Sku);
                    if (null != statusInfo)
                    {
                        viewModel.Status = statusInfo.Status;
                        viewModel.Case = statusInfo.Case;
                        viewModel.UserPrepInstructions = statusInfo.UserPrepInstructions;
                        viewModel.UserNote = statusInfo.UserNote;
                        viewModel.IsNoteImportant = statusInfo.IsNoteImportant ?? false;
                        if (statusInfo.MissingDimensions ?? false)
```

```
                    {
                        if (!skuMissingDimension.Contains(item.Sku))
                        {
                            skuMissingDimension.Add(item.Sku);
                        }
                    }
                }
                if (String.IsNullOrEmpty(viewModel.Title))
                {
                    var title = missingSkuTitles.SingleOrDefault(m => m.Sku ==
viewModel.Sku)?.ProductName;
                    if (!string.IsNullOrEmpty(title))
                    {
                        viewModel.Title = title;
                    }
                }
                shipmentListViewModel.Add(viewModel);
            }
listOfData.AddRange(EntityExtensions.ConvertDbShipmentToExcelDtWithVariation(f
baInboundShipment, shipmentListViewModel));
        }
            foreach (var fbaInboundShipmentItems in
fbaInboundShipments.Select(shipment =>
_sellermogulContext.FbaInboundShipmentItems.Where(i =>
i.FbaInboundShipmentId.Equals(shipment.Id,
StringComparison.OrdinalIgnoreCase))).ToList())
            {
                foreach (var partOffbaInboundShipmentItems in
fbaInboundShipmentItems.Partition(10))
                {
```

```
                    var responses = await ExponentialBackoffFunction.AsyncOperation(
                        () => aws.ItemLookup(partOffbaInboundShipmentItems.Select(a
=> a.Asin).ToArray(),
                        ItemLookupRequestIdType.ASIN,
                        AwsResponseGroupType.ItemAttributes,
                        SearchIndex.All),
                        1500);
                    foreach (var item in responses.SelectMany(response =>
response.Item))
                    {
                        asinUpcs.Add(item.ItemAttributes != null
                            ? new AsinUpcsModel { Asin = item.ASIN, Upcs =
item.ItemAttributes.UPCList }
                            : new AsinUpcsModel { Asin = item.ASIN, Upcs = null });
                    }
                }
            }
            var allBoxes = GetBoxes(collectionId);
            var shipmentId = fbaInboundShipments.Select(s => s.Id).ToList();
            var destinations = await
_sellermogulContext.FbaDestinationAddresses.Where(d =>
shipmentId.Contains(d.ShipmentId)).ToListAsync();
            Dictionary<string, string> destinationAddresses = new Dictionary<string,
string>();
            if (fbaCollection.DeliverrPlanId == null)
            {
                destinations.ForEach(d => { destinationAddresses.Add(d.ShipmentId,
$"FBA:
{shipFrom.CompanyName.ToUpper()}\n{d.Name}\n{d.AddressLine1}\n{(!String.IsNu
```

```
llOrEmpty(d.AddressLine2) ? d.AddressLine2 + "\n" : String.Empty)}{d.City},
{d.StateOrProvinceCode} {d.PostalCode}\nUnited States"); });
        }
            else
            {
               destinations.ForEach(d => { destinationAddresses.Add(d.ShipmentId,
$"{d.Name.Substring(0, 3)} c/o
Deliverr\n{d.AddressLine1}\n{(!String.IsNullOrEmpty(d.AddressLine2) ?
d.AddressLine2 + "\n" : String.Empty)}{d.City}, {d.StateOrProvinceCode}
{d.PostalCode}\nUS"); });
        }
            return new UnlimitedJsonResult
            {
               Data = newShipmentCollectionData
               {
                  Shipments = listOfData,
                  Boxes = allBoxes,
        Destinations = destinationAddresses
               }
            };
         }
         catch (Exception exception)
         {
            return Json(new { Success = false, Message = exception.Message },
JsonRequestBehavior.AllowGet);
        }
        }
        public async Task<JsonResult> AddOrUpdateBox(Box model)
        {
         try
```

```
        {
            if (string.IsNullOrEmpty(model.ShipmentId) || model.CollectionId ==
Guid.Empty) return Json(new { Success = false, Message = "Invalid model." },
JsonRequestBehavior.AllowGet);
            if (model.Id == null || model.Id == Guid.Empty)
            {
                var allParcels =
                    await _sellermogulContext.FbaInboundParcels.
                    Where(p => p.CollectionId == model.CollectionId && p.ShipmentId
== model.ShipmentId).
                    OrderBy(p => p.ParcelIndex).
                    Select(p => (int)p.ParcelIndex).ToListAsync();
                int newParcelNumber = 1;
                if (allParcels.Count > 0)
                {
                    var max = allParcels.Max() + 1;
                    newParcelNumber = Enumerable.Range(1,
max).Except(allParcels).Min();
                }
                model.ParcelIndex = newParcelNumber;
                model.Id = Guid.NewGuid();
                model.IsOpened = true;
                model.BoxNumber = string.Empty;
                model.FbaInboundParcelItems = null;
                model.FbaInboundShipment = null;
                model.FbaShipmentCollection = null;
             model.UserId = UserId;
                _sellermogulContext.FbaInboundParcels.AddOrUpdate(model);
            }
            else
```

```
            {
                if (model.ParcelIndex <= 0) return Json(new { Success = false, Message =
"Invalid box number." }, JsonRequestBehavior.AllowGet);
                var dbModel = await
_sellermogulContext.FbaInboundParcels.FirstOrDefaultAsync(p => p.Id == model.Id);
                if (null == dbModel)
                    return Json(new { Success = false, Message = "Invalid box." },
JsonRequestBehavior.AllowGet);
                dbModel.CopyFromWithParam(model, new string[] { "ShipmentId",
"ParcelIndex", "FbaInboundParcelItems", "FbaInboundShipment",
"FbaShipmentCollection" });
                dbModel.BoxNumber = string.Empty;
                model.FbaInboundParcelItems = null;
                model.FbaInboundShipment = null;
                model.FbaShipmentCollection = null;
                model.BoxNumber = String.Empty;
                _sellermogulContext.FbaInboundParcels.AddOrUpdate(dbModel);
            }
            await _sellermogulContext.SaveChangesAsync();
            WriteToLog("AddOrUpdateBox", model);
            return Json(new { Success = true, Message = model },
JsonRequestBehavior.AllowGet);
        }
        catch (Exception exception)
        {
            WriteExceptionToLog("AddOrUpdateBox", model, exception);
            return Json(new { Success = false, Message = exception.Message },
JsonRequestBehavior.AllowGet);
    }
    }
```

```
        public async Task<JsonResult> DeleteBox(FbaInboundParcel model)
        {
            try
            {
                var dbModels = new List<FbaInboundParcel>();
                foreach (var model in models)
                {
                    var dbModel = await
_sellermogulContext.FbaInboundParcels.FirstOrDefaultAsync(p => p.Id == model.Id);
                    if (null == dbModel)
                        return Json(new { Success = true, Message = "Box " +
model.ParcelIndex + " (" + model.ShipmentId + ") not found in the DB." },
JsonRequestBehavior.AllowGet);
                    if (IsFakeParcel(model))
                        RemoveFakeParcel(dbModel);
                    else
                    {
                        dbModels.Add(dbModel);
                        await RemoveMerchantsInBox(dbModel);
                        //restore lot
                        if (!String.IsNullOrEmpty(dbModel.ShipmentId) &&
dbModel.ShipmentId != "!M!" && !String.IsNullOrEmpty(dbModel.MerchantLot))
                        {
                            var lot =
_sellermogulContext.FbaInboundScannedMerchants.Single(m => m.Lot ==
dbModel.MerchantLot);
                            var lotItem = _sellermogulContext.FbaInboundParcelItems.First(pi
=> pi.ParcelId == dbModel.Id && pi.Sku == lot.Sku);
                            if (lot.PendingQty != null)
                            {
```

```
                    lot.PendingQty = lot.PendingQty + (int)lotItem.Scanned;
                }
                else
                {
                    lot.Qty = lot.Qty + (int)lotItem.Scanned;
                    var lotHistory =
_sellermogulContext.FbaInboundScannedMerchantsHistories.Single(h =>
h.FbaInboundScannedMerchantLot == lot.Lot && h.Type ==
FbaInboundScannedMerchantsHistory.OperationType.Outbound && h.Qty == -
(int)lotItem.Scanned);

_sellermogulContext.FbaInboundScannedMerchantsHistories.Remove(lotHistory);
                }

_sellermogulContext.FbaInboundScannedMerchants.AddOrUpdate(lot);
                }
            }
        }
        var parcelsItems =
_sellermogulContext.FbaInboundParcelItems.SqlQuery("Select * From
FbaInboundParcelItems " + SqlExtensions.SqlInOperation(models.Select(s =>
s.Id.ToString()), "ParcelId")).ToList();
        if (parcelsItems.Count > 0)

_sellermogulContext.FbaInboundParcelItems.RemoveRange(parcelsItems);
        if (dbModels.Count > 0)
            _sellermogulContext.FbaInboundParcels.RemoveRange(dbModels);
        await _sellermogulContext.SaveChangesAsync();
        WriteToLog("DeleteBoxes", models);
```

```
            return Json(new { Success = true, Message = "Removed" },
JsonRequestBehavior.AllowGet);
        }
        catch (Exception exception)
        {
            WriteExceptionToLog("DeleteBoxes", models, exception);
            return Json(new { Success = false, Message = exception.Message },
JsonRequestBehavior.AllowGet); } }
    public async Task<JsonResult> AddOrUpdateBoxItem(ParcelItemparcelItem)
    {
    try
        {
    if (parcelItem.Scanned<= 0 || string.IsNullOrEmpty(parcelItem.Sku) ||
parcelItem.ParcelId == null || parcelItem.ParcelId == Guid.Empty)
    throw new Exception("Invalid model.");
            var parcel = parcelItem.FbaInboundParcel;
            if (parcelItem.FbaInboundParcel?.ShipmentId == "!M!")
            {
                parcelItem.FbaInboundParcel.ShipmentId = null;
                parcelItem.FbaInboundParcel.MerchantLot = await
MerchantWarehouseExtension.GenerateMerchantLot(GetSiteSettingValueAsync,
SetSiteSettingValueAsync);
                parcelItem.FbaInboundParcel.FbaInboundScannedMerchants = new
List<FbaInboundScannedMerchant>
                {
                    new FbaInboundScannedMerchant
                    {
                        Lot = parcelItem.FbaInboundParcel.MerchantLot,
                        Date = DateTime.Now,
                        FbaInboundParcelId = parcel.Id,
```

```
                    Qty = (int)parcelItem.Scanned,
                    Sku = parcelItem.Sku,
                    Redestribution = true,
                    FbaInboundScannedMerchantsHistory = new
List<FbaInboundScannedMerchantsHistory>
                    {
                        new FbaInboundScannedMerchantsHistory
                        {
                            Date = DateTime.Now,
                            FbaInboundScannedMerchantLot =
parcelItem.FbaInboundParcel.MerchantLot,
                            Id = Guid.NewGuid(),
                            Qty = (int)parcelItem.Scanned,
                            Type =
FbaInboundScannedMerchantsHistory.OperationType.Inbound
                        }
                    }
                }
            };
        }
        var isFakeParcel = parcelItem.FbaInboundParcel == null ? false :
IsFakeParcel(parcelItem.FbaInboundParcel);
        var parcelExists = await
_sellermogulContext.FbaInboundParcels.AnyAsync(p => p.Id == parcelItem.ParcelId);
        if (!parcelExists)
            isFakeParcel = true;
        if (isFakeParcel)
        {
            parcel = parcelItem.FbaInboundParcel;
            parcel.FbaInboundParcelItems = null;
```

```
                parcel.UserId = UserId;
                parcel.IsOpened = !closeParcel;
                _sellermogulContext.FbaInboundParcels.Add(parcel);
                RemoveFakeParcel(parcel);
                await _sellermogulContext.SaveChangesAsync();
            }
            else if (closeParcel)
            {
                var dbParcel = await
_sellermogulContext.FbaInboundParcels.FirstOrDefaultAsync(p => p.Id ==
parcelItem.ParcelId);
                if (null == dbParcel)
                    throw new Exception("Invalid parcel.");
                dbParcel.IsOpened = false;
                _sellermogulContext.FbaInboundParcels.AddOrUpdate(dbParcel);
            }
            parcelItem.LastUpdate = DateTime.Now.ToUniversalTime();
            parcelItem.FbaInboundParcel = null;
            if (parcelItem.Id == null || parcelItem.Id == Guid.Empty)
            {
                parcelItem.Id = Guid.NewGuid();
                parcelItem.ParcelId = parcel.Id;
                _sellermogulContext.FbaInboundParcelItems.AddOrUpdate(parcelItem);
                if (!String.IsNullOrEmpty(parcel.ShipmentId) && parcel.ShipmentId !=
"!M!" && !String.IsNullOrEmpty(parcel.MerchantLot))
                {
                    var lot = _sellermogulContext.FbaInboundScannedMerchants.Single(m
=> m.Lot == parcel.MerchantLot);
                    if (lot.PendingQty != null && lot.PendingQty > 0)
                    {
```

```
                    if (lot.PendingQty >= (int)parcelItem.Scanned)
                    {
                        lot.PendingQty = lot.PendingQty - (int)parcelItem.Scanned;
                    }
                    else
                    {
                        lot.PendingQty = 0;
                    }
                }
                else
                {
                    lot.Qty = lot.Qty - (int)parcelItem.Scanned;
                    var lotHistory = new FbaInboundScannedMerchantsHistory
                    {
                        Date = DateTime.Now,
                        FbaInboundScannedMerchantLot = parcel.MerchantLot,
                        Id = Guid.NewGuid(),
                        Qty = -(int)parcelItem.Scanned,
                        Type =
FbaInboundScannedMerchantsHistory.OperationType.Outbound
                    };
                    _sellermogulContext.FbaInboundScannedMerchantsHistories
.AddOrUpdate(lotHistory);
                }
            _sellermogulContext.FbaInboundScannedMerchants.AddOrUpdate(lot);
            }
        }
        else
        {
```

```
                var dbModel = await
_sellermogulContext.FbaInboundParcelItems.FirstOrDefaultAsync(p => p.Id ==
parcelItem.Id);
                if (null == dbModel)
                    throw new Exception("Invalid Box Item.");
                dbModel.CopyFromWithParam(parcelItem, new string[] { "Sku",
"ParcelId", "Parcel", "FbaInboundParcel" });
                _sellermogulContext.FbaInboundParcelItems.AddOrUpdate(dbModel);
            }
            await _sellermogulContext.SaveChangesAsync();
            if (parcel != null)
            {
                parcel.FbaInboundParcelItems =
parcel.FbaInboundParcelItems.ClearCollectionCircularReferences();
                parcel.FbaInboundScannedMerchants = null;
            }
            WriteToLog("AddOrUpdateBoxItem", new { parcelItem =
parcelItem.ClearCircularReferences(), closeParcel = closeParcel });
            return Json(
                new
                {
                    Success = true,
                    BoxModel = parcel,
                    BoxItemModel = parcelItem.ClearCircularReferences(),
                    NeedBoxUpdate = isFakeParcel
                }, JsonRequestBehavior.AllowGet);
        }
        catch (Exception exception)
        {
```

```
            WriteExceptionToLog("AddOrUpdateBoxItem", new { parcelItem =
parcelItem.ClearCircularReferences(), closeParcel = closeParcel }, exception);
            return Json(new { Success = false, Message = exception.Message },
JsonRequestBehavior.AllowGet); } }
```

Для видалення з коробки товару, потрібно викликати метод DeteleBoxItem контролера FbaInbound, передавши аргументом екземпляр класу ParcelItem. Код метода є наступним:

```
public async Task<JsonResult> DeleteBoxItem(ParcelItem model)
{
   try
   {
      var dbModel = await
_sellermogulContext.FbaInboundParcelItems.FirstOrDefaultAsync(p => p.Id ==
model.Id);
      if (null == dbModel)
      {
         WriteToLog("DeleteBoxItem", new { model = model, collectionId =
collectionId });
         return Json(new { Success = true, Message = "Box Item not found in the
DB." }, JsonRequestBehavior.AllowGet);
      }
      _sellermogulContext.FbaInboundParcelItems.Remove(dbModel);
      _sellermogulContext.SaveChanges();
      WriteToLog("DeleteBoxItem", new { model = model, collectionId =
collectionId });
      return Json(new { Success = true, Message = model },
JsonRequestBehavior.AllowGet);
   }
   catch (Exception exception)
   {
```

```
            WriteExceptionToLog("DeleteBoxItem", new { model = model, collectionId
= collectionId }, exception);
            return Json(new { Success = false, Message = exception.Message },
JsonRequestBehavior.AllowGet); } }

public JsonResult VoidTransportRequest(Guid shipmentCollectionId, string
shipmentId)
{
   try
   {
      var fbaMws = new FbaInboundServiceMws(MwsLoginModel);

      var shipmentIds = new string[] { };
      if (string.IsNullOrEmpty(shipmentId))
      {
         shipmentIds = _sellermogulContext.FbaInboundShipments.Where(s =>
s.FbaShipmentCollectionId == shipmentCollectionId).Select(s => s.Id).ToArray();
      }
      else
         shipmentIds = _sellermogulContext.FbaInboundShipments.Where(s =>
s.FbaShipmentCollectionId == shipmentCollectionId && s.Id == shipmentId).Select(s
=> s.Id).ToArray();

      var dbShipmentsList =
_sellermogulContext.FbaInboundShipments.SqlQuery("Select * From
FbaInboundShipments " + SqlExtensions.SqlInOperation(shipmentIds, "Id")).ToList();

      foreach (var _shipmentId in shipmentIds)
      {
```

```
            var voidTransportRequest =
fbaMws.InvokeVoidTransportRequest(_shipmentId);
            if (voidTransportRequest == null ||
!voidTransportRequest.IsSetVoidTransportRequestResult()) return Json(new { Success
= false, Message = "Something wrong in InvokeVoidTransportRequest shipmentId: " +
shipmentId }, JsonRequestBehavior.AllowGet);
            if
(voidTransportRequest.VoidTransportRequestResult.TransportResult.IsSetErrorDescrip
tion()) return Json(new { Success = false, Message =
StringExtensions.CreateErrorJsonMessage(_shipmentId,
voidTransportRequest.VoidTransportRequestResult.TransportResult.ErrorDescription)
});

            var shipment = dbShipmentsList.FirstOrDefault(s => s.Id == _shipmentId);
            if (shipment != null)
            {
                shipment.TransportStatus = (int)TransportStatuses.Voided;
                _sellermogulContext.FbaInboundShipments.AddOrUpdate(shipment);
            }
        }
        _sellermogulContext.SaveChanges();
        return Json(new { Success = true, Message = "Ok" },
JsonRequestBehavior.AllowGet);
    }
    catch (FBAInboundServiceMWSException exception)
    {
        ForceCleanTransportStatus(shipmentCollectionId, shipmentId);
        return Json(new { Success = false, Message = exception.Message, Source =
"amazon" }, JsonRequestBehavior.AllowGet);
    }
```