Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Система автоматизації обліку матеріальних цінностей на складі

Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ЗППІПЗ. 180106.01.04.ПЗ

Виконав студент IV курсу група ІПЗ-18-1 _____ О. Ю. Лучицький
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ Н. І. Праворська
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер ст. викладач _____ Г.І. Бедратюк
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

1 червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
01 03 2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Лучицькому Олегу Юрійовичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Система автоматизації обліку матеріальних цінностей на складі

Керівник проєкту (роботи) Праворська Наталія Іванівна
кандидат педагогічних наук, доцент

Затверджена наказом ректора університету від 01.03.2022 р. № 18

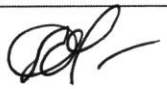
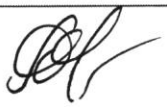
2. Строк подання студентом проєкту (роботи) на кафедру 01.06.2022 р.

3. Вихідні дані до проєкту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди, 20 шт.)

6. Консультанти розділів дипломного проєкту (роботи)

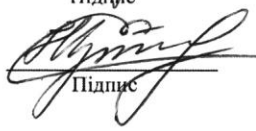
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Бедратюк Г.І., ст. викладач		
Антиплагіат	Гурман І.В., доцент кафедри ІПЗ		

7. Дата видачі завдання «01» березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем ДП	01.12– 30.12.2021	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2022	
3 Проектування програмного забезпечення	01.02 – 28.02 2022	
4 Програмна реалізація	01.03 – 10.04.2022	
5 Тестування програмного забезпечення	11.04 – 30.04.2022	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2022	
7 Попередній захист ДП	травень 2022 (згідно графіка)	
8 Перевірка ДП на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2022	
9 Підготовка до захисту та захист ДП	з 01.06.2022	

Студент  О. Ю. Лучицький
Підпис Ініціали, прізвище

Керівник проекту (роботи)  Н. І. Праворська
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: «Система автоматизації обліку матеріальних цінностей на складі».

Автор проекту: Лучицький Олег Юрійович.

Керівник проекту: Праворська Наталія Іванівна.

Пояснювальна записка: 128 с., 25 рис., 5 табл., 3 дод., 18 джерел.

Графічна частина: 20 презентаційних слайдів.

ДОДАТОК, ПРОГРАМНИЙ ДОДАТОК, БАЗА ДАНИХ, БД, СИСТЕМА УПРАВЛЕННЯ БАЗАМИ ДАНИХ, СУБД, C#, .NET FRAMEWORK, WINDOWS FORMS, MICROSOFT VISUAL STUDIO, MICROSOFT SQL SERVER.

Метою проекту є розробка програмного забезпечення для обліку матеріальних цінностей, яка надає доступ до бази даних магазину комп'ютерної техніки та дозволяє виконання операцій редагування бази через засоби інтерфейсу.

У дипломному проекті досліджено специфіка складу та складського обліку, досліджено найвідоміші та найпоширеніші рішення у області складського обліку з визначенням їх переваг та недоліків та функціоналу. Був проведений аналіз існуючих архітектур баз данхи та систем, які з ними взаємодіють. На основі отриманої інформації було розроблено програмне забезпечення на основі локальної бази даних

Для реалізації програмного забезпечення використано мову програмування C# та платформу розробки .NET Framework, бібліотеку інтерфейсів Windows Forms, інтегроване середовище розробки Microsoft Visual Studio 2019 та базу даних Microsoft SQL Server Express 2016 LocalDB.

В результаті було виконана розробка локальної системи керування бази даних для операційної системи Windows.

30.05.2021р.
Дата

О. Луч
Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ЗППІПЗ. 180106.01.04.ПЗ	Пояснювальна записка	128		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні	20		
			слайди			

Змн.	Арк.	№ докум.	Підпис	Дата	ЗППІПЗ. 180106.01.04.ПЗ			
Виконав		Лучицький О.Ю.		30.05.	Система автоматизації обліку матеріальних цінностей на складі	Літ.	Арк.	Аркушів
Керівник		Праворська Н.К.		30.05.			1	1
Н. контр.		Бедратюк Г. І.		30.05.	Відомість документів	ХНУ, ІПЗ-18-1		
Зав. каф.		Бедратюк Л.П.		30.05.				

ЗМІСТ

ВСТУП

Сучасні тенденції в області підприємницької діяльності відображають значне поширення технологій в сфері електронної інформації та автоматизації процесів. Окрім цього, на сьогодення взаємодія з інформацією (доступ, знаходження, збереження) пов'язана з діяльністю не тільки спеціалістів в області інформаційних технологій, але й багатьох інших людей. Перехід на оцифровувану інформацію сприяє економії у витратах для працівників підприємства у процесі взаємодії (пошуку, отримання, збереження, збуту, обміну) з ресурсами – як матеріальними, так і нематеріальними.

У підприємств, які ще досі використовують паперовий документообіг, з часом виникають труднощі у звернені до власної інформаційної системи. Це пов'язано з тим, що протягом довгого часу існування, бухгалтерія такого підприємства накопичує величезні об'єми документації, перегляд якої стає необхідним для успішного пошуку, незважаючи на те, скільки та які документи буде попередньо переглянуто без успіху. І навіть при цьому успіх може бути негарантованим. На додачу ще є ризики втрат документів по ряду причин, починаючи з крадіжки та закінчуючи природнім старінням паперових носіїв.

Така ситуації створила в сфері підприємницької діяльності попит на засоби оптимізації процесу бухгалтерського обліку, які б забезпечили більш швидке використання інформації з точним результатом роботи та могли б скомпонувати величезні приміщення для зберігання паперів у більш доступну організовану структуру електронної системи. Такий засіб виник з розвитком інформаційних та комп'ютерних технологій.

Про своєчасність та актуальність розглянутої проблеми говорить той факт, що більшу частину свого часу завідуючий складом та відділ закупівель витрачають на оформлення різної документації та звітів. А також те, що, не дивлячись на сучасну поширеність програмних засобів ведення обліку, значна

частина підприємств – в основному малих та фізичних осіб-підприємців – досі використовують традиційні методи обліку.

Метою проекту є розробка програмного забезпечення для обліку матеріальних цінностей, яка надає доступ до бази даних магазину комп'ютерної техніки та дозволяє виконання операцій редагування бази через засоби інтерфейсу.

Завданнями роботи є:

– дослідити процес проведення обліку матеріальних цінностей під час процесу роботи складського відділу;

– проаналізувати основні програмні продукти, які вже розроблені та використовуються підприємствами на сьогоднішній день, порівняти їхні переваги та недоліки, визначити необхідний набір функцій, який має пропонувати ПЗ даного типу;

– розробити архітектуру проектованої системи, вибрати комплекс технології та інструментів для подальшого процесу розробки;

– реалізувати систему, яка задовольняє потрібний функціонал;

– виконати тестування розробленої системи на предмет знаходження недоліків та помилок реалізації.

Результатом дипломного проекту має бути робоче програмне забезпечення, яке надає можливість швидкого введення, виведення та пошуку інформації, яка стосується матеріальних цінностей на складі.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Сутності складу дано численні визначення, але всі вони, здебільшого, містять багато спільних рис у своїх характеристиках.

Таким чином, склад – комплекс одного або декількох нежитлових приміщень, технічних та організаційних засобів, задача яких полягає у збереженні різних вантажів з дотриманням умов зберігання кожного для подальшого їх розподілу та використання споживачем.

Також існує визначення складу, як засобу керування резервами на різних етапах матеріального потоку.

Структура і кількість складських приміщень визначаються типом, масштабом і номенклатурою вантажів та їх матеріаломісткістю. Складські приміщення мають дуже розвинуту систему класифікації, яка включає численні ознаки.

Склади розділяють за різновидами продукції, яка знаходиться на зберіганні:

– склад матеріалів;

– сировини;

– інструментів;

– обладнання та запасних частин;

– готової продукції;

– бракованої продукції або відходів виробництва.

Склади розділяють за різноманіттям продукції на зберіганні:

– універсальні – склади, на яких можна зберігати кілька видів вантажу (центральний матеріальний склад);

– спеціалізовані – використовуються для зберігання однорідної продукції (склади для окремих матеріалів, однотипної продукції тощо);

Склади розділяють за функціональним призначенням:

– довгострокового збереження;

– перевалочні;

– розподільчі;

– сезонного збереження;

– митні.

Поділяють за специфікою конструювання приміщення:

– закриті (складські будівлі);

– напівзакриті (навіси);

– відкриті (резервуари, елеватори і т. п).

Класифікація відбувається за багатьма іншими ознаками: за напрямом в логістиці, учасниками в системі логістики, формою власності, масштабом, транспортною інфраструктурою тощо.

Ще один спосіб класифікації стоїть окремо – класифікація за класами. В цій класифікації розглядається сукупність характеристик складського приміщення, дослідження яких визначає відповідність приміщення вимогам, які забезпечують найкраще виконання функцій складу. В якості зразку розглянуто міжнародну класифікацію від Knight Frank, яка має розповсюдження і на території України. Вона включає шість класів: A+, A, B+, B, C, D. Черговість вказана за спаданням якості складського приміщення. Тобто до приміщень класу А відносять такі, що побудовані з матеріалів та за вимогами, необхідними для складу, а до класу D – фактично будь-яке приміщення, яке підприємство визначило як складське, без додаткових вимог.

Діяльність складу готової продукції ніколи не пов’язана з комерційною діяльністю. Загальний набір робіт, який виконують на різних складах, практично немає відмінностей. Це зумовлено схожістю функцій, які виконує склад під час різних операцій з логістики. Однак можливі відмінності в

діяльності окремого складу, які пов’язані з його структурними особливостями та призначенням.

Функціонал складу включає в себе діяльність щодо прийому матеріальних цінностей, їх збереження, обліку та відпуску споживачеві:

– прийом та збереження вантажу;

– виконання операцій, що передують видачі продукції та/або сировини у виробництво;

– видача продукції та/або сировини у виробництво згідно з визначеним порядком;

– виконання операцій, що передують збуту вантажу споживачеві (комплектування, етикетування, пакування тощо);

– збут вантажу споживачу;

– здійснення обліку руху резервів(запасів) та їх регулювання;

– здійснення діяльності, напрямленої на модернізацію діяльності в області складського господарства;

– формування звітності про діяльність складського відділу та участь у розгляді скарг на діяльність підприємства.

Складський облік – це сортовий, кількісний оперативно-технічний облік, веденням якого займаються матеріально-відповідальні особи складського господарства. Здійснюється облік на складі за допомогою книги складського обліку.

Відповідальним за облік запасів на виробництві призначають завідуючого складу (комірника), який є матеріально-відповідальною особою.

Основними задачами складського обліку становлять:

– забезпечення цілісності та контролю за рухом і правильним використанням всіх матеріальних цінностей;

– дотримання норм запасів та витрат;

– вчасне відстежування матеріалів, які не використовуються;

– отримання точної інформації про залишки на складі.

Залежно від кількості номенклатури та асортименту товару, складський облік можна вести такими способами як:

– номенклатурний або картковий – одиниці номенклатури відповідає одна картка. В основному застосовується при невеликій номенклатурі без розділення на різні групи;

– партіонний – ведення обліку партіями за термінами придатності, за постачальником;

– сортовий – розподілення товару по групах, підгрупах, типах, сортах;

– партійно-сортовий – такий метод передбачає, що після прибуття товару на склад, його приймання відбувається партією, а потім розподіляється чи комплектується окремими групами за схожими характеристиками. Підходить для великої кількості номенклатури та товарного асортименту.

Основне завдання впровадження систем складського обліку – покращення складських процесів шляхом підвищення їх швидкості та точності, що призводить більш ефективного ведення бізнесу.

Специфікою розроблюваного програмного забезпечення є спеціалізація для, в основному, «молодих» малих підприємств та фізичних осіб-підприємців, облік для котрих проходить по спрощеним процедурам і характеризується використанням традиційних методологій ведення обліку. В таких випадках, об’єм облікових робіт ще не виступає настільки критичним параметром діяльності, що для підвищення ефективності стає необхідним використання інформаційних технологій.

Однак, враховуючи значне поширення комп’ютерної техніки, впровадження простої системи автоматизації в діяльності може бути доцільним і, до того ж, обов’язковим для малого підприємця, який планує розширювати та розвивати свою діяльність.

Проблемою в цій ситуації виступає відсутність поширених програмних продуктів в цій досить, специфічній сфері. Переважна більшість облікових систем розраховані на досить розвинутого підприємця, і потребують як значних

грошових витрат, так і досвіту та певних знань в сфері бухгалтерського обліку.

З іншого боку, наявність невеликого простого програмного додатку, дозволяє швидко почати його застосування під час роботи підприємства без значних витрат у часі на опанування. До того ж, такий додаток буде досить дешевим, що позитивно впливає на його доступність. Окрім цього, таке програмне забезпечення не потребує особливих вимог для початку використання, що робить ще більш універсальним та, зокрема, дозволяє використовувати на операційних системах, які вважаються застарілими, що теж не рідкість серед малих підприємств.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Впровадження засобів автоматизації бухгалтерського обліку характеризується помітним позитивним впливом на ефективність процесів, які протікають на підприємстві. Тому використання програмної продукції стає, фактично, безумовною складовою більшості підприємницьких процесів.

Такий попит вказав на перспективи значного прибутку за рахунок продажу програмного забезпечення та подальшого його супроводу. Подібні можливості провокують конкуренцію розробників програмного забезпечення, які пропонують різноманітні рішення з власними особливостями. Таке різноманіття, очікувано, змушує підприємця задумуватися на предмет пошуку програмного продукту з найкращим для нього співвідношенням ціни та функціоналу, який надає програма.

Далі буде досліджено кілька програмних засобів, які надають функції в області складського обліку. Для початку розглянемо лінійку продукції «BAS», яку буде представляти продукт «BAS: Бухгалтерія».

Створена, як альтернатива російській «1С: Підприємство» та отримавши поширення після накладання санкцій на російський продукт, «BAS», фактично,

має однаковий набір функцій і повторює філософію пропонування різних рішень, спрямованих на різні області автоматизації бізнес-процесів.

«BAS: Бухгалтерія» забезпечує облік у наступних сферах:

– облік декількох організацій з різними системами оподаткування в одній базі;

– облік запасів, облік торгових операцій, облік грошових рахунків;

– облік розрахунків з контрагентами;

– облік основних запасів, малоцінних необоротних матеріальних активів, нематеріальних активів и малоцінних швидкозношуваних предметів;

– облік виробництва;

– облік податок на додану вартість;

– облік зарплати;

– закриття періодів, звітів.

Стосовно питання щодо складського обліку, розглянуте програмне забезпечення проводить, як кількісний, так і кількісно-сумовий облік. Також до функціоналу включена можливість проведення інвентаризації товару, в результаті якої, можливе визначення та списування нестачі і знаходження надлишку.

Система надає можливість відображення як постачання, так і збуту товарів. У випадку роботи у роздрібній торгівлі, програмний продукт надає контроль безпосередньо продажу та аналізувати процесу продажу після здійснення інвентаризації.

Виробник вказує на наступні особливості свого продукту:

– інтерфейс «Таксі», за допомогою якого користувач здатен налаштовувати відображення програми відповідно до власних бажань (рисунок 1.1);

– зручність додавання даних, команд та документів до «Вибраного» за допомогою одного кліка;

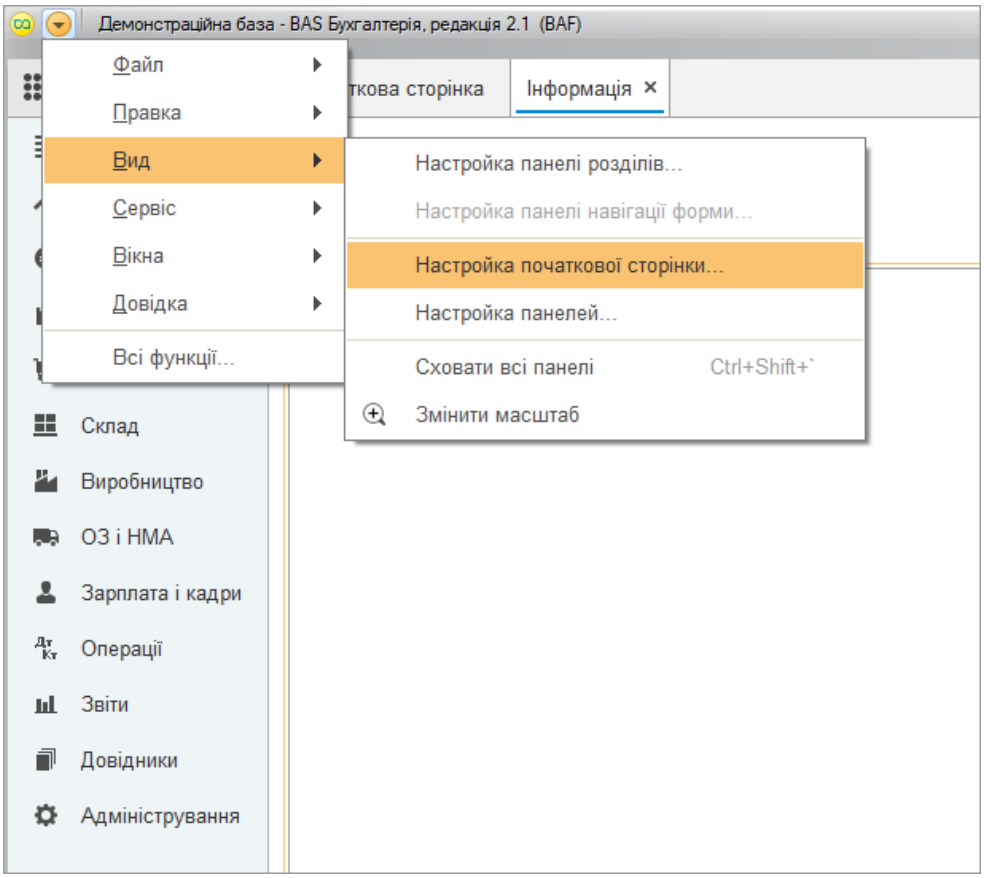


Рисунок 1.1 – Налаштування вікна «BAS: Бухгалтерія» через інтерфейс «Таксі»

– зручний пошук у «Вибраному» багатофункціональним управлінням, текстовим пошуком та можливістю перейменовувати елементи;

– комфортність ведення обліку різних компаній завдяки єдиній інформаційній базі;

– керівники компанії можуть оцінювати матеріальні та виробничі запаси за допомогою урахування середньої собівартості, відповідно з ціною продажу, з урахуванням власної ідентифікованої собівартості.

Також «BAS: Бухгалтерія» пропонує особливості роботи користувачів з різними ролями (посадами).

Таким чином, за допомогою закладки «Керівнику», програмний продукт надає керівнику спеціальний інструмент для формування звітності. В цьому розділі зібрано звіти, найбільш затребувані керівником у вигляді таблиць і діаграм (рисунок 1.2 і рисунок 1.3).

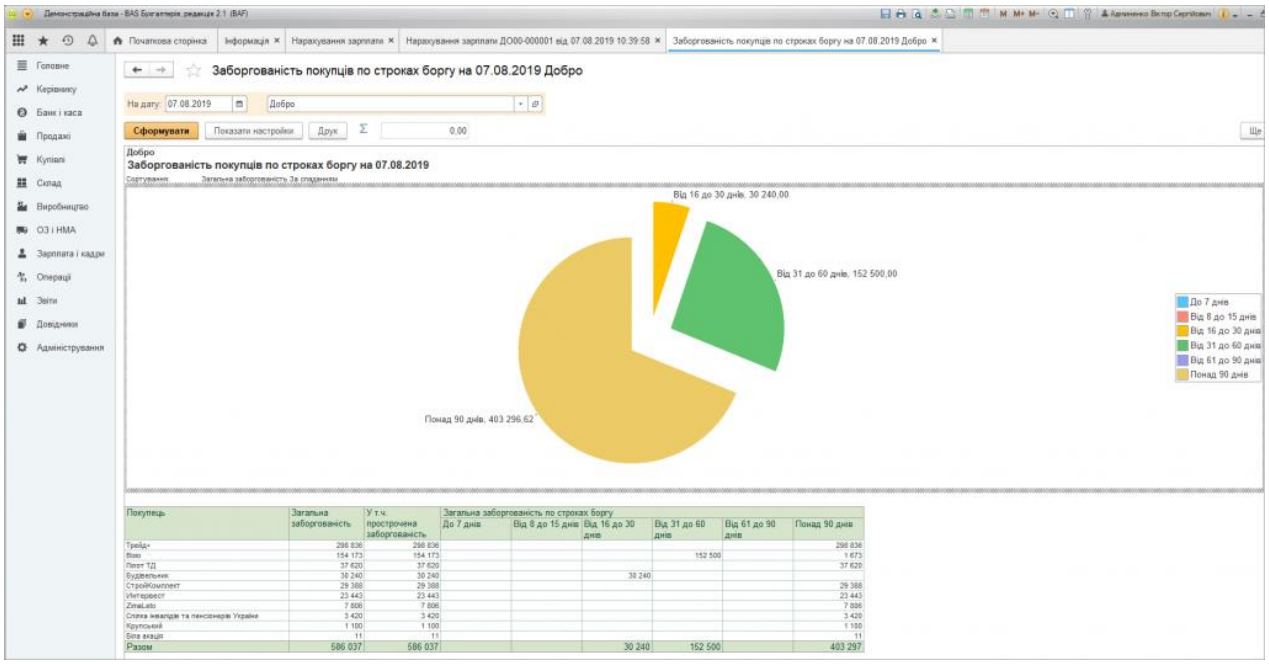
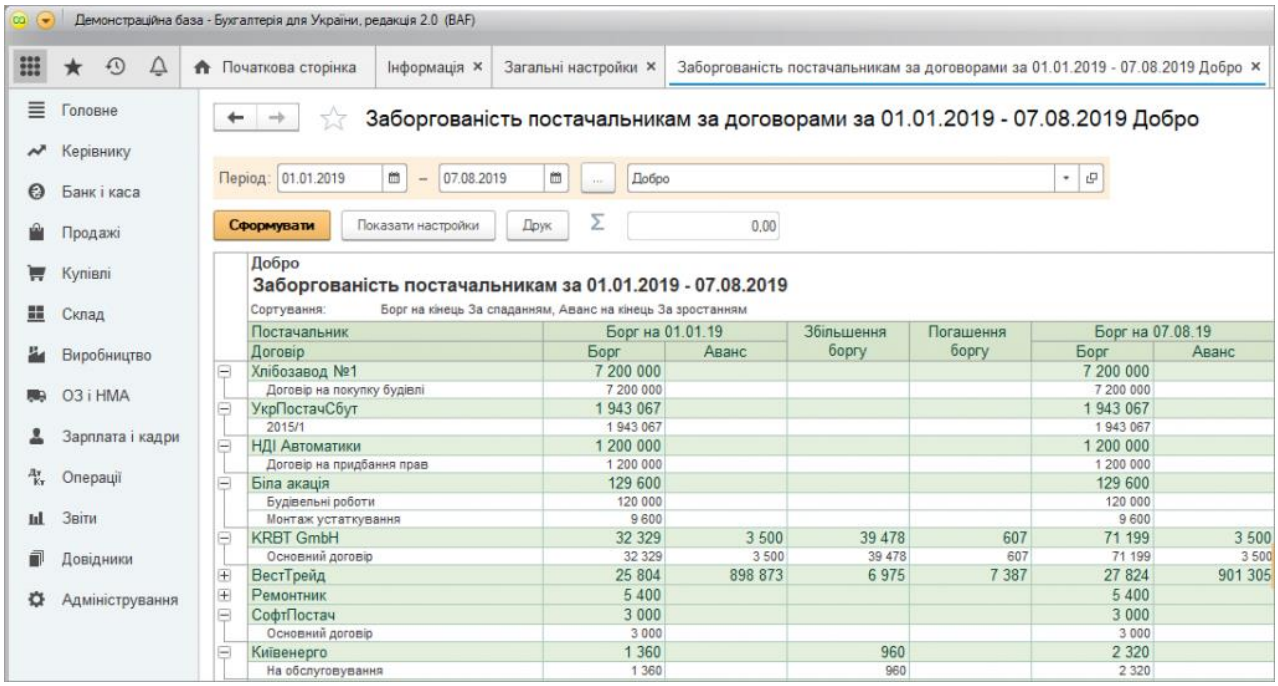


Рисунок 1.2 – Надання інформації керівнику через інтерфейс «BAS: Бухгалтерія»



Постачальник / Договір	Борг на 01.01.19		Збільшення боргу	Погашення боргу	Борг на 07.08.19	
	Борг	Аванс			Борг	Аванс
Хлібозавод №1	7 200 000				7 200 000	
Договір на покупку будівлі	7 200 000				7 200 000	
УкрПостачСбут	1 943 067				1 943 067	
2015/1	1 943 067				1 943 067	
НДІ Автоматики	1 200 000				1 200 000	
Договір на придбання прав	1 200 000				1 200 000	
Біла акація	129 600				129 600	
Будівельні роботи	120 000				120 000	
Монтаж устаткування	9 600				9 600	
KRBT GmbH	32 329	3 500	39 478	607	71 199	3 500
Основний договір	32 329	3 500	39 478	607	71 199	3 500
ВестТрейд	25 804	898 873	6 975	7 387	27 824	901 305
Ремонтник	5 400				5 400	
СофтПостач	3 000				3 000	
Основний договір	3 000				3 000	
Київенерго	1 360		960		2 320	
На обслуговування	1 360		960		2 320	

Рисунок 1.3 – Надання інформації керівнику через інтерфейс «BAS: Бухгалтерія»

Будучи, по факту, ідейним спадкоємцем «1С» на новій платформі розробки, рішення «BAS» унаслідували і його переваги, до яких можна віднести:

– різноманітний набір рішень дозволяє обрати програмний продукт з набором функцій, які підходять для конкретного виду підприємства з конкретним розміром персоналу;

– забезпечення регулярних оновлень, що надає підтримку актуальності функцій, які пропонує програмний продукт;

– завдяки договору, який формується на основі придбання програмного продукту, користувач забезпечується технічною підтримкою;

– об’єктно-орієнтоване середовище розробки BAF надає високий рівень стандартизації розробки, внаслідок чого, стає можливим швидко впроваджувати нові технологічні рішення;

– платформа дозволяє, при потребі, зміну коду продукту;

– самостійним «плюсом» програмного продукту є впровадження мережевого спілкування: чати, відеоконференції.

Аналогічно до переваг, спостерігається й наслідування недоліків, які можливо охарактеризувати як фундаментальні, оскільки вони пов’язані напряму з перевагами, які пропонує програмний продукт:

– для використання програмного продукту, розробник пропонує ліцензію, яка має немалу ціну та тимчасова, тобто, потребує додаткових постійних грошових витрат;

– регулярні оновлення, не дивлячись на свою користь, можуть призвести до ситуації, коли підприємство, залишившись однієї з основних складових власної діяльності, фактично, стає недієздатним на певний час;

– одне рішення може не покривати своїми можливостями вимоги підприємства, що призводить до придбання кількох продуктів та додаткових витрат;

– як наслідок передньої ситуації, користувач може мати справу з надмірним набором функцій, якими на ділі не користується і за котрі переплачує;

– маючи як універсальні, так більш спеціалізовані рішення, «BAS» не здатен покривати специфічні потреби конкретного підприємства, які доводиться додавати самотужки, що також витратно;

– інтеграція продукту на підприємстві потребує платні послуги спеціалістів;

– ризик виникнення помилок в роботі програми після оновлень;

– поширення українських санкцій на продукцію «BAS» забороняє її застосуванням в державних установах України, однак заборона не поширюється на діяльність приватних підприємств.

Наступним програмним продуктом було розглянуто «Торгсофт». Це програмний продукт для автоматизації в області роздрібної торгівлі від однойменної української компанії-розробника. Програма, по аналогії з описаною вище «BAS: Бухгалтерія», є комплексною. Функціонал продукту поширюється на області, які можуть бути пов’язані з обліком: товарознавство, складські документи, облік фінансів, складський облік, торгові звіти, аналіз торгівлі, маркетинг тощо.

У питанні складського обліку виділяється наступний функціонал:

– комплектація і розукомплектація товару;

– додавання товарів у відомість інвентаризації;

– перелік товарів на складі з прив’язкою до дерева видів товарів;

– перелік та аналіз моделей товару на складі;

– відстежування руху товару;

– зворотна відомість за кількістю та собівартістю;

– аналіз оборотності запасів;

– список оприбуткувань;

– перелік витрат;

– сума товару за період;

– реєстр прибуткових накладних;

– журнал складських документів;

– зміна ціни продажу;

– розрахунок роздрібних цін по націнці та курсу;

– показ накладних та товарів, які знаходяться в дорозі;

– показ накладних та товарів після внутрішньої передачі;

– облік фіскального і нефіскального товару на складі, для контролю залишків фіскального товару;

– гарантійний облік;

– перегляд кількості товарів у всіх магазинах торговельної мережі;

– синхронізація з інтернет-магазином;

– відвантажування товару особисто клієнту з віддаленого складу;

– інформаційно-пошукова система, яка дозволяє швидко знайти товар на складі;

– реєстр внутрішніх передач;

– реєстр складських документів;

– облік скасування продавцем товару з реалізації;

– облік обладнання і техніки, які використовуються на підприємстві;

– розрахунок роздрібних і гуртових цін по націнці та курсу;

– валютний та регіональний прайс-листи;

– розрахунок складських запасів;

– показ середньої націнки на торгових точках, з урахуванням товарних знижок на складі та можливістю видрукувати звіт.

До переваг «Торгсофт» можна віднести:

– наявність ліцензії (на відміну від продуктів, схожих на «1С» та «BAS») без обмежень в часі, що значно зменшує грошові витрати;

– використання нового програмного продукт легше засвоїти за рахунок зосередження функціоналу на приватного підприємця та роздрібну торгівлю;

– менші вимоги до користувача в питанні знань бухгалтерської сфери;

– програма надає широкий функціонал в своїй спеціалізації (будучи вже готовим, сформований рішенням), дозволяє легко розрахувати витрати;

– в програмі об’єднані функції як локального, так і мережевого додатку;

– можливість роботи з торгівельним обладнанням;

– підтримка оновлень;

– наявність технічної підтримки.

Також можна виділити й недоліки:

– спеціалізація вказує на можливу непридатність задовільнити вимоги підприємств інших видів;

– сформована програма не дозволяє, за необхідності, додавання специфічних функцій для конкретного підприємства;

– можлива проблема перевантаження функціоналом;

– спроба не роз’єднувати локальну та мережеву версії приводить до проблем у роботі з віддаленим сервером;

– технічна підтримка платна;

– для баз даних, з якими працює програма, розробниками не передбачені засоби захисту.

Приклад інтерфейсу програми продемонстровано на рисунку 1.4.

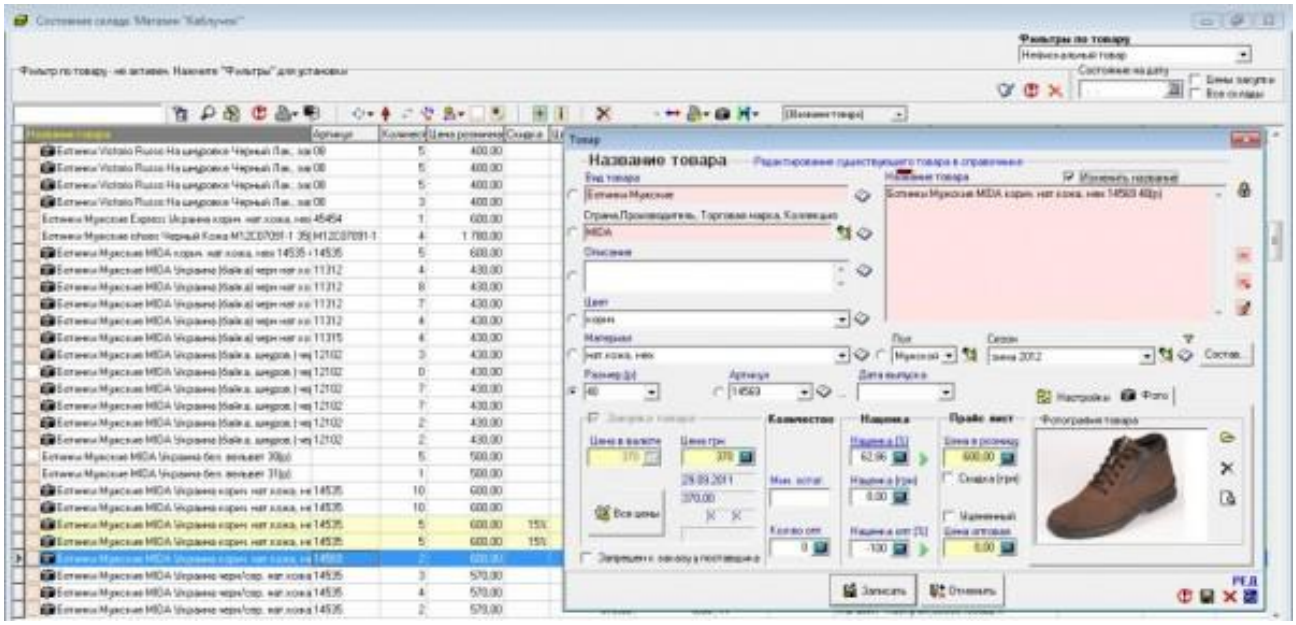


Рисунок 1.4 – інтерфейс програми «Торгсофт»

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання

Після аналізу предметної області та існуючих програмних рішень можна визначити та описати вимоги, передбаченні до розроблюваного програмного забезпечення. Для цього буде застосована мова UML (англ. Unified Modeling Language) – уніфікована мова моделювання, яку використовують для візуального відображення різноманітних систем: як програмних додатків так і бізнес-систем.

При використанні мови UML усі елементи системи відображаються у вигляді спеціальних графічних конструкцій, які отримали назву діаграм.

Для опису розроблюваного програмного забезпечення використовуються діаграма варіантів використання – найпростіша діаграма UML, яка відображує взаємодію між акторами (користувачами) та, власне, варіантами використання функцій системи. Вона застосовується для опису функціональних вимог до програми та формує загальну представлення того, як програма може бути застосована. Сформовано характеристику користувачів (акторів) та функціоналу, який надано їм (варіанти використання). Вони представлені у таблиці 1.1 та таблиці 1.2.

Таблиця 1.1 – Опис акторів розроблюваного ПЗ

Актор	Короткий опис
1	2
Користувач	Виконує авторизацію всередині системи. Може через систему переглядати інформацію або здійснювати її зміну (додавання, видалення, зміну) а також виводити її у вигляді файлу PDF.
База даних	Хоча повноцінним актором не являється, є зовнішнім елементом програмного додатку. Через базу даних проходять запити на зміну або відображення

Продовження таблиці 1.1

1	2
	інформації, які вона виконує.

Таблиця 1.2 – Опис варіантів використання розроблюваного програмного забезпечення

Актор	Найменування варіанту використання	Опис варіанту використання
1	2	3
Користувач	Авторизація	Користувач отримує доступ до програми через введення логіну та паролю.
	Вибір типу записів	Користувач обирає, з записами якої категорії взаємодіяти.
	Зчитування інформації	Користувач відсилає до бази даних запит, у відповідь до якої база даних має відобразити у інтерфейсі свої записи.
	Зміна інформації	Користувач відсилає до бази даних запит стосовно додавання нового запису, видалення або зміни існуючого. У відповідь база даних виконує відповідну дію.
	Ввести інформацію про запис	Користувач зобов’язаний ввести інформацію, яка буде у новому записі або в оновленому.
	Створити PDF файл	Користувач через діалогове вікно файлової системи обирає місце

Продовження таблиці 1.2

1	2	3
		зберігання файлу формату PDF, в якому буде записана одна з таблиць бази даних.
База даних	Зчитування інформації	База даних відображає записи, на які користувач відправив запит.
	Зміна інформації	База даних виконує додавання нового запису, видалення або зміни існуючого у відповідності до вхідного запиту.

Визначивши акторів системи та варіанти використання, побудуємо діаграму варіантів використання (рисунок 1.5).

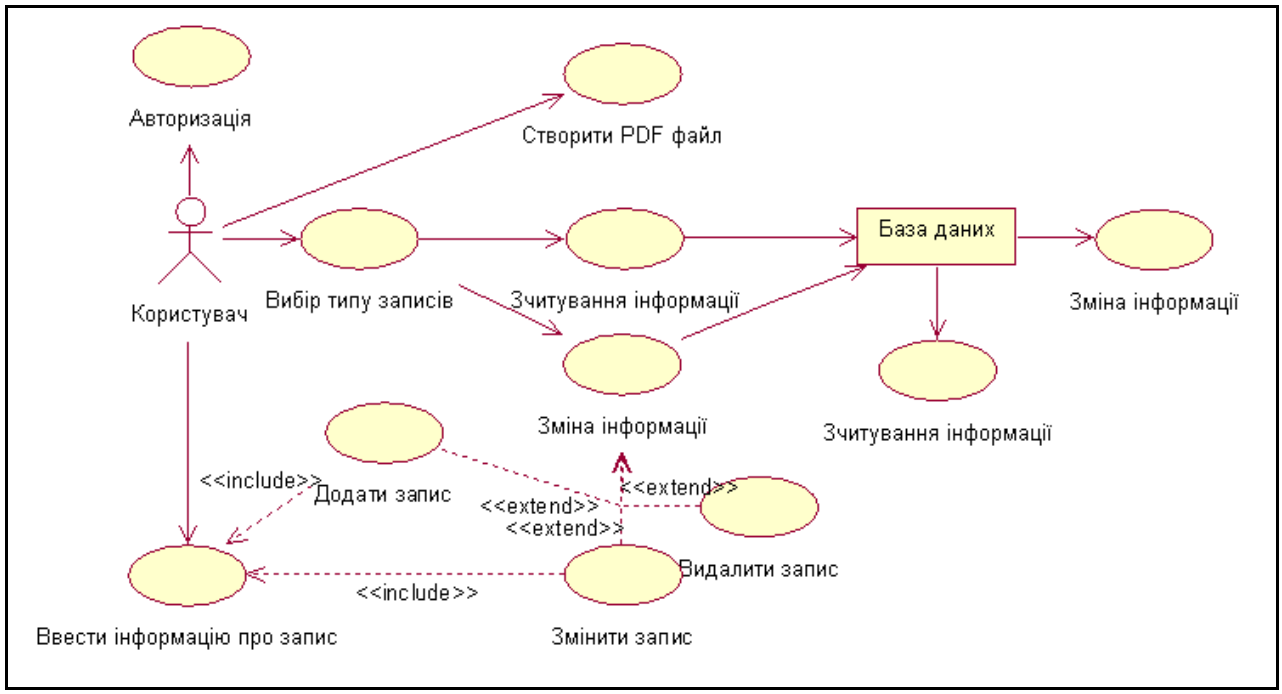


Рисунок 1.5 – Діаграма варіантів використання розроблюваного ПЗ

В результаті визначення вимог до розроблюваного програмного забезпечення вимог На основі проведеного аналізу вимог до ПЗ було розроблене технічне завдання, яке подано у додатку А.

У даному розділі було проведено аналіз та огляд діяльності складу та складського обліку. Були розглянуті приклади існуючого програмного забезпечення в області складського обліку, їх можливості, особливості, переваги та недоліки. На основі отриманої інформації відбулося формування вимог до розроблюваного програмного забезпечення та створено технічне завдання.

2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕНЯ

2.1 Аналіз та проектування архітектури системи

Одним з головних етапів розробки програмного забезпечення є створення архітектури. Правильно розроблена архітектура забезпечує якісне розуміння логіки роботи програмного забезпечення в цілому та його окремих компонентів, що полегшує її подальшу розробку, а згодом, і підтримку. Розробити універсальну розгорнуту архітектуру дуже складно, якщо взагалі можливо. Архітектура програмного забезпечення сильно залежить від конкретних завдань, які були визначені розробниками.

Архітектуру програмного забезпечення визначено різними способами. Але їх можна звести до двох категорій:

– структура взаємопов’язаних елементів;

– певний набір рішень та принципів, за якими виконується створення програмного забезпечення.

Для подальшої роботи використано об’єднане поняття. Архітектура програмного забезпечення – це комплекс структурних елементів, їх взаємозв’язків та уявлень, що визначають основні властивості розроблюваного програмного забезпечення.

Архітектура систем, які працюють з базами, залежать від самих баз даних. Розділяють два види баз даних:

– централізовані;

– розподілені.

Централізовані бази даних зберігаються на одному комп’ютері, до якої можна отримати доступ через інший, якщо даний комп’ютер знаходиться в одній мережі з ним. Таке характерно для систем які використовуються в локальній мережі підприємства.

Розподілені бази даних характеризуються поділом на декілька частин, які розподілено по між комп’ютерами в одній мережі.

За способом доступу розділяють бази даних з локальним доступом, та мережевим. Останні розділяють за типами доступу:

– «файл-сервер»;

– «клієнт-сервер».

У системі «файл-сервер» користувачі виконують запуск програми на головній машині з самою базою даних, після чого на комп’ютері користувача відкривається її копія та копія головної бази даних, яка оновлюється разом з основною. Логіка програми відбувається також на локальній машині. Така структура характерна для мереж з невеликою кількістю користувачів, оскільки її ефективність падає при одночасній спробі кількох користувачів отримати доступ до однакових даних.

У системі «клієнт-сервер» головна машина зберігає і базу даних, і систему управління нею. При цьому на комп’ютерах користувачів не відбувається робота безпосередньо з базою даних, а тільки формують запити, які опрацьовуються на сервері.

Враховуючи поставлену задачу, в якості архітектури було обрана архітектура локальної системи керування базою даних для одного користувача. Архітектуру розділено на три рівні:

– інтерфейс – графічне відображення програмного додатку та інформації, яку надає база даних, для користувача. Представляє можливість авторизації для початку роботи. Дозволяє за допомогою елементів графічного оформлення введення інформації та виклик функцій взаємодії з базою даних, відображає реакцію програми на дії користувача;

– система управління базою даних – внутрішня реалізація функцій взаємодії з базою даних;

– база даних – джерело даних, до яких через програмний додаток реалізацію доступу користувача з подальшою модифікацією.

Специфікою даної архітектури є те, що поділ інтерфейсу та керування базою даних умовне, оскільки обробка запиту відбувається напряму в процесі обробки взаємодії з інтерфейсом. Графічне представлення архітектури представлена нижче на рисунку 2.1.

Рисунок 2.1. – Графічне представлення розроблюваного програмного забезпеченння

Взаємодія з записами представлена чотирьома діями:

– перегляд записів;

– додавання нового запису на основі попередньо введеної інформації;

– оновлення існуючого запису на основі попередньо введеної інформації;

– видалення існуючого запису.

Інтерфейс сформовано засобами Windows Forms, в якому кожна форма – набір графічних елементів інтерфейсу – представлена як окремий клас, з власними описами реакцій на взаємодію користувача з елементами інтерфейсу.

Розглянемо взаємодію користувача з розроблюваним програмним забезпеченням. Для цього можна використати діаграми послідовності, за допомогою яких графічно зображується взаємодія об’єктів системи між собою у

часі (послідовності). Спершу – авторизація користувача перед початком роботи. Користувач запускає програму, яка виводить форму для введення логіну та паролю. При здійснені введення відбувається перетворення вхідних даних у хеш-функцію, яка вже перевіряється з контрольними даними. При збігу вхідних і контрольних даних, надається доступ до основних форм програми. У разі неправильного введення з’явиться відповідне повідомлення. Дана процедура з успішним результатом графічно представлена на рисунку 2.2.

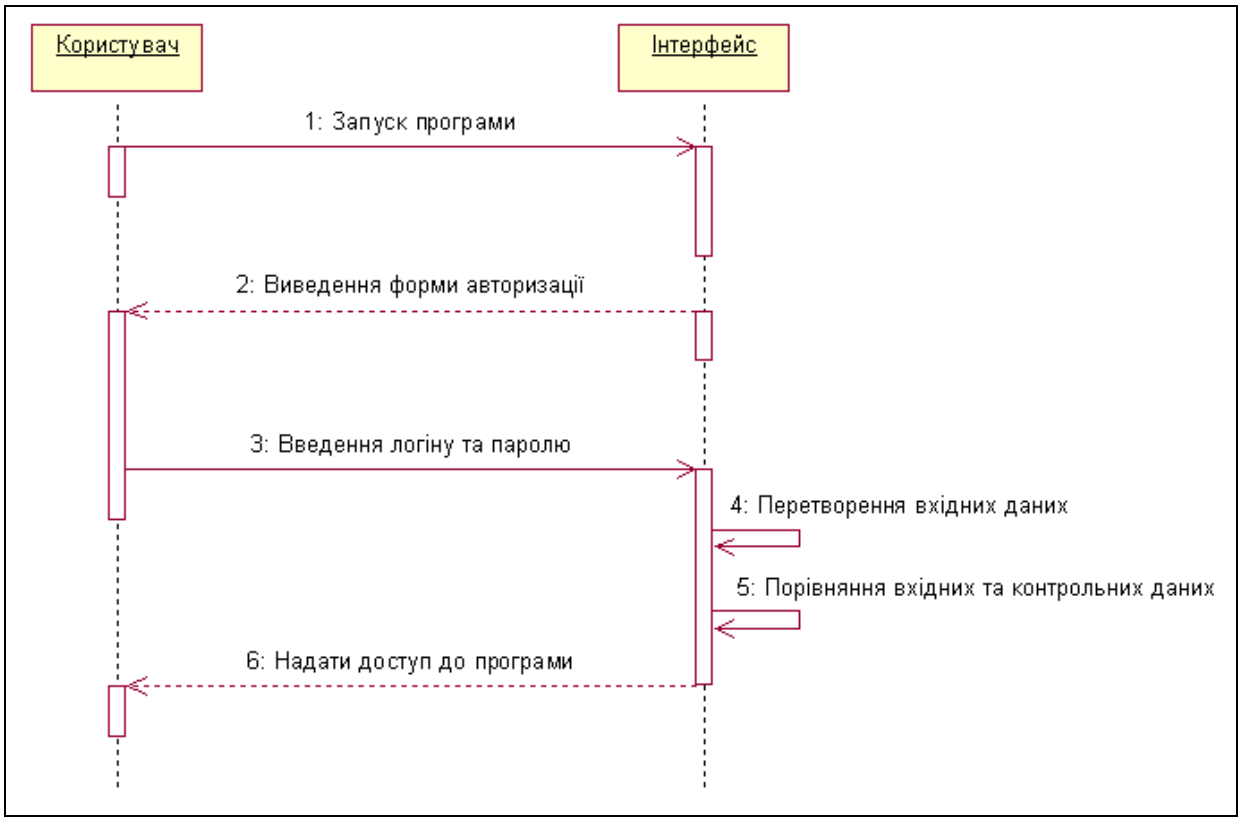


Рисунок 2.2. – Діаграма послідовності авторизації користувача

Далі процедура авторизації в діаграмах буде опущена для зручності.

Для введення нового запису буде передбачена наступна послідовність: користувач вибирає через інтерфейс відповідний режим редагування бази даних. Інтерфейс змінюється для надання доступу до відповідних функцій, після чого користувач вводить вхідні дані і надсилає запит. У разі успіху база даних утворює новий запис, а інтерфейс виводить відповідне повідомлення.

Аналогічно у випадку невдачі. Дана процедура з успішним результатом графічно представлена на рисунку 2.3.

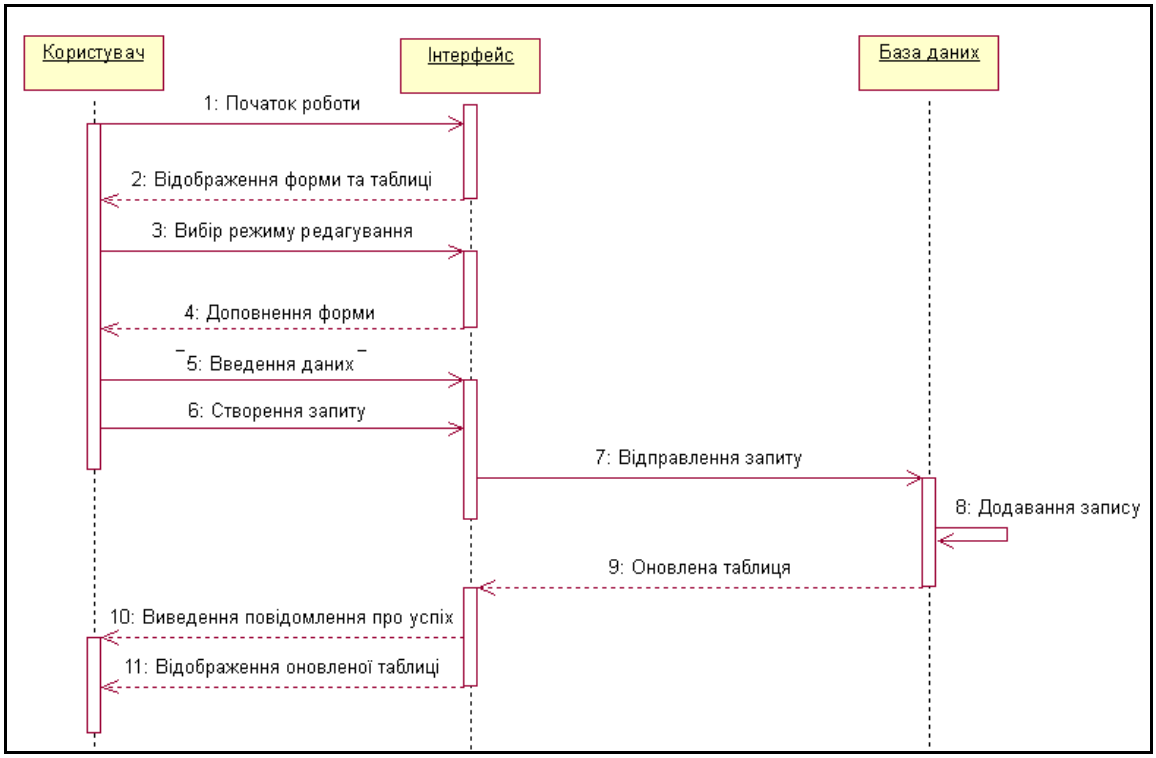


Рисунок 2.3. – Діаграма послідовності введення нового запису

Для видалення запису з бази даних послідовність дій аналогічна попередній, але після оновлення форми інтерфейсом, користувач обирає з таблиці записів цільовий і вже потім створює запит з аналогічною реакцією від бази даних. Дана процедура з успішним результатом графічно представлена на рисунку 2.4.

Для оновлення запису з бази даних послідовність дій аналогічна діям при додаванні запису, але після оновлення форми інтерфейсом, користувач обирає з таблиці записів цільовий і змінює необхідні йому дані. Після цього створює запит з аналогічною реакцією від бази даних.

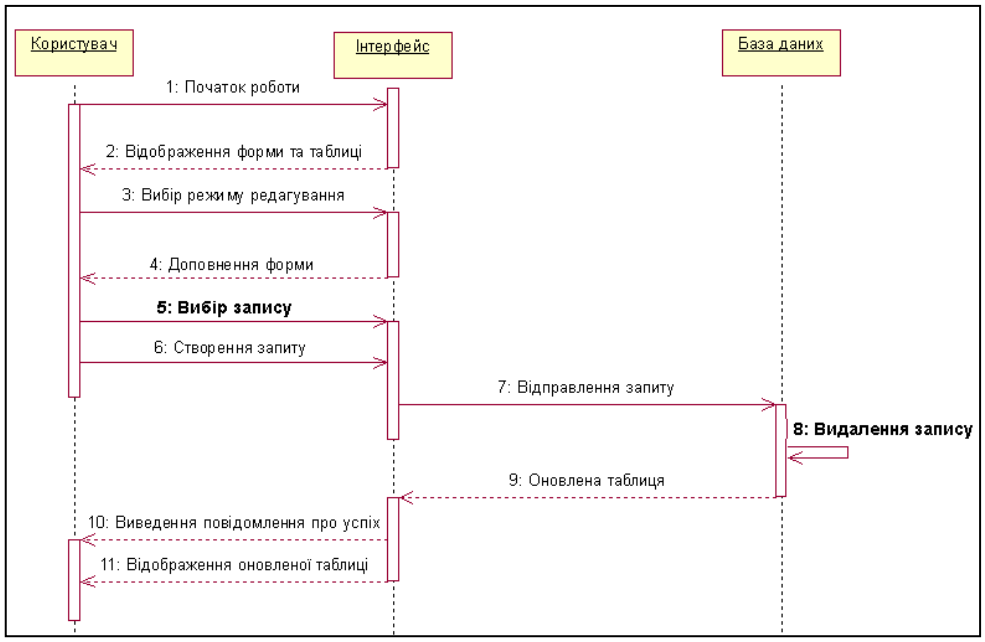


Рисунок 2.4. – Діаграма послідовності видалення запису (новий елемент виділено напівжирним)

Дана процедура з успішним результатом графічно представлена на рисунку 2.5.

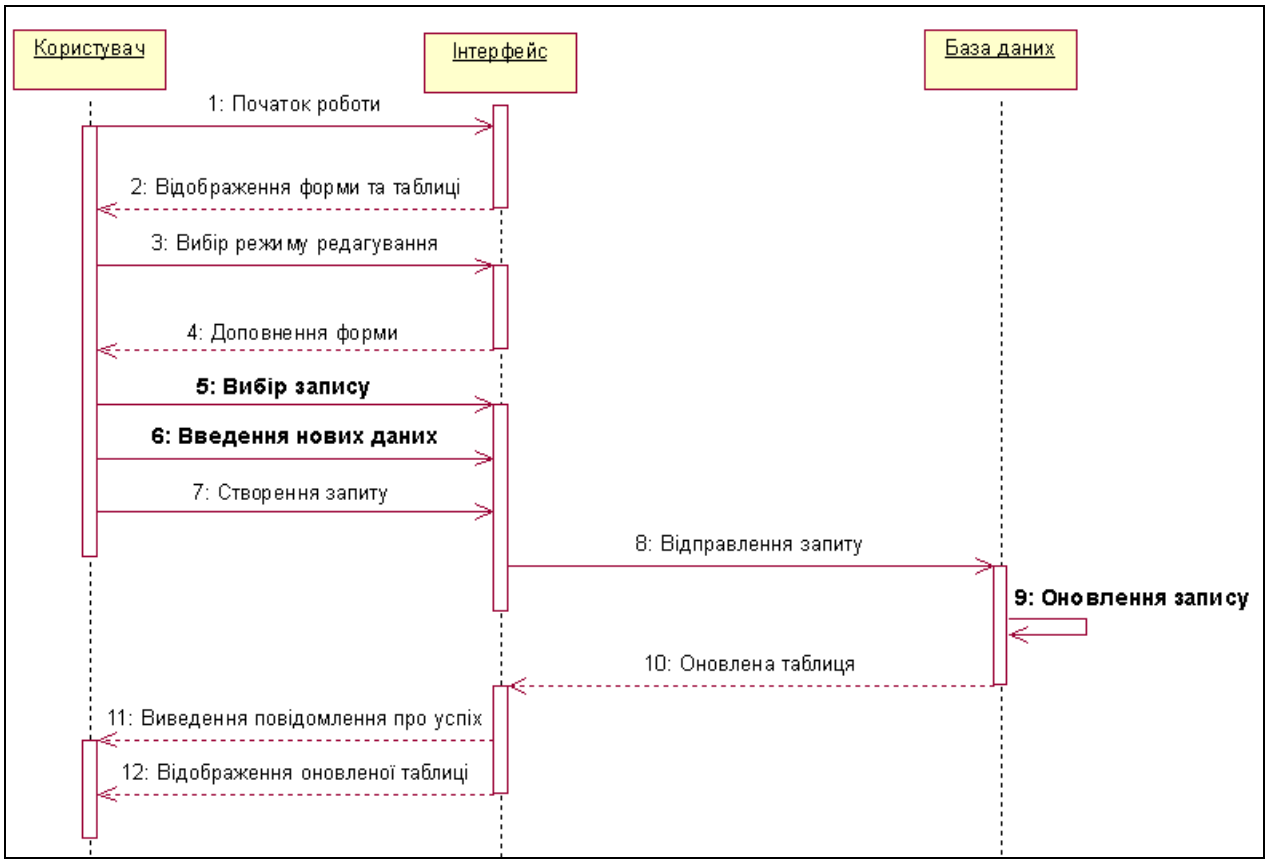


Рисунок 2.5. – Діаграма послідовності оновлення запису (новий елемент виділено напівжирним)

Останньою дією буде створення PDF-файлу з введенням обраних табличних даних. У цьому сценарії, повторюються дії, як в попередній послідовності до кроку з вибором режиму редагування. Після відображення потрібної таблиці користувач відправляє запит через інтерфейс до системи файлового діалогу Windows. Після вказання місця збереження та назви файлу, у разі успіху, діалогове вікно закривається без повідомлень. Дана процедура з успішним результатом графічно представлена на рисунку 2.6.

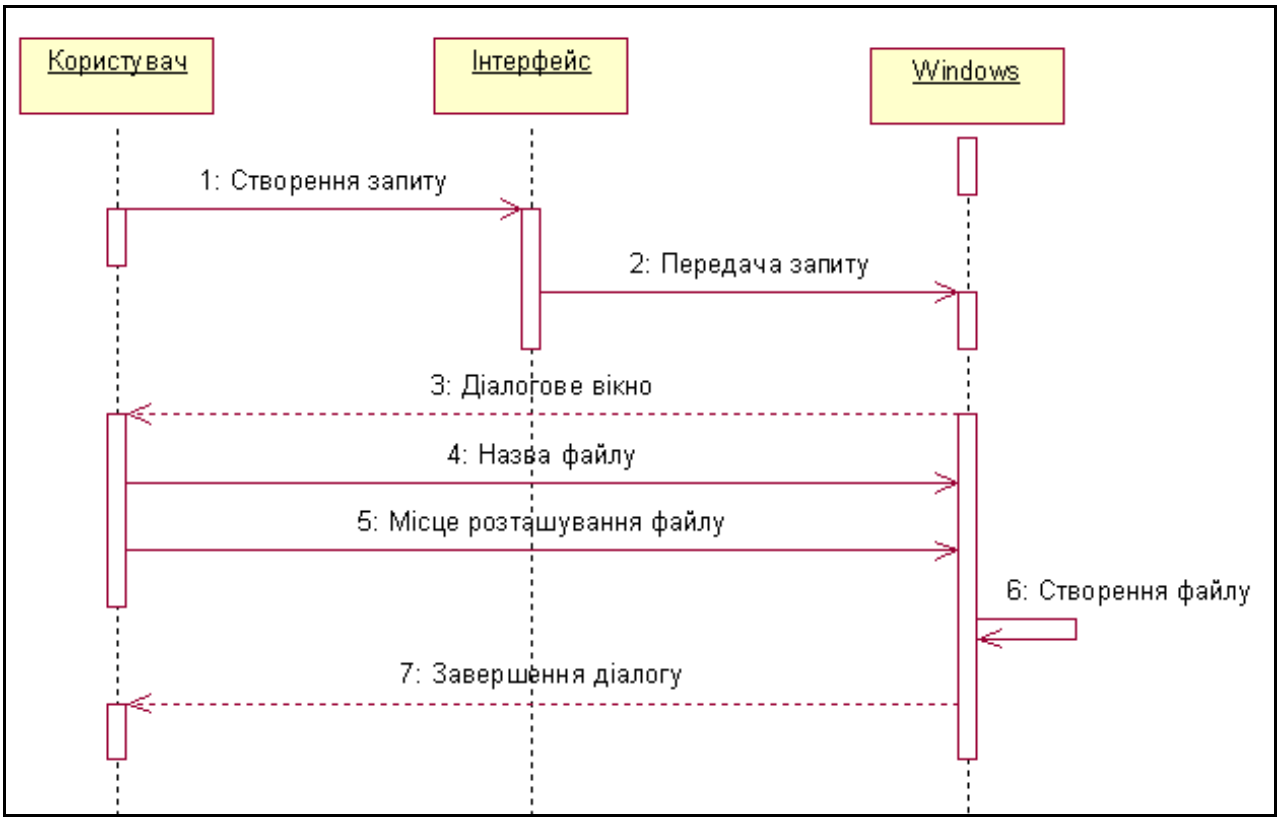


Рисунок 2.6. – Діаграма послідовності утворення PDF-файлу

2.2 Аналіз та вибір типу бази даних, проектування структури бази даних

Вибір бази даних потребує дослідження їх різноманітних видів для вибору та послідуючого впровадження в проект. Основним критерієм класифікації розглянутих баз даних виступає модель даних – незмінна система організації та

збереження інформації в базі даних. Були розглянуті ієрархічні, мережеві та реляційні бази даних

Ієрархічна модель історично представляє собою першу модель даних. Ця модель представлена у вигляді деревоподібної структури, яка розходиться гілками донизу, слідуючи ієрархічної логіки побудови. На сьогоднішній день основне своє застосування знаходять у формуванні інформуванні інформаційних структур у форматі XML, для організації роботи з даними в мережі Інтернет. А також яскравим прикладом ієрархічної бази даних є деревоподібна файлова система на комп'ютерах.

Ієрархічна модель характеризується обов'язковим створенням батьківського і дочірнього об'єктів, які формують наступні складові структури:

– атрибут – мінімальна одиниця інформації, яка описує властивість конкретного об'єкта;

– запис – сукупність атрибутів, яка утворює екземпляр описуваного об'єкта;

– групове відношення – взаємозв'язок між різними записами, що визначає взаємодію батьківської і дочірньої записів;

– ключовий елемент – спеціальний атрибут, який формується з унікальних даних для кожного запису.

Взаємозв'язки між об'єктами характеризуються тим, що у кожного об'єкта верхнього рівня може бути деяка кількість спадкових об'єктів, але кожен об'єкт-спадкоємець може походити тільки від одного об'єкта. Тобто у ієрархічній моделі можливі тільки зв'язки типу «один до одного» і «один до багатьох». Формування зв'язку «багато до багатьох» можливе тільки при дублюванні об'єктів, узгодження даних між якими неможливе.

До переваг виділяють ідеальне застосуванні при роботі з ієрархічно впорядкованими даними з раціональним використанням пам'яті та чудовою швидкодією.

Недоліками такої моделі даних становлять відсутність гнучкості, прямого доступу до даних, неможливість наявності в одного об’єкта кількох батьків та пошуку з нижній рівнів ієрархії догори.

У мережевій моделі даних ролі об’єктів отримали деяке розширення. Будь який об’єкт може бути і головним, і підлеглим. Один об’єкт здатен містити властивості обох ролей в наборі даних. З цього виходить що один об’єкт може мати необмежену кількість взаємозв’язків.

Так само, як і ієрархічна модель, мережева модель може бути зображена як орієнтований граф, за винятком того, що граф може мати циклічні конструкції, що вказують на наявність у підлеглої вершини декілька головних. Така конструкція виділяється більшою гнучкістю і придатністю до виконання більш широкого діапазону завдань.

Мережева база даних, як і ієрархічна, має навігацію. Це означає, що отримання доступу до даних по шляхах, не передбачених при створенні бази даних, може потребувати нерозумно тривалого часу. Така технологія дозволяє оптимізацію та прискорення роботи з даними, але це виливається у підвищення залежності від структури програм і самих даних.

Програми обробки даних в такій ситуації стають жорстко прив’язаними до конкретної структури бази даних і потребують переписування у разі змін. Крім того, принцип навігації не дозволяє зробити процес маніпулювання записаною інформацією доступним для непрофесіональних користувачів. У процесі знаходження цільового запису, програміст мусить визначати шлях, по якому потім необхідно буде переглядати всі наявні записи.

Усе це, вказує на значні трудові витрати при розробці засобів практичного використання вище вказаних типів баз даних. Складність практичної реалізації баз даних на основі ієрархічної і мережної моделей визначила створення реляційної моделі даних.

Бази на основі реляційних моделей використовують двовимірні таблиці, які складаються з деякої кількості стовпців (атрибутів), що містять інформацію про певні параметри. В такій таблиці одному стовпчику буде відповідати

інформація одного типу – числа, текст тощо. Записи такої бази даних утворюються через рядок (кортеж) в таблиці, який створює фіксований набір атрибутів. Послідовність кортежів у таблиці визначається не порядком їх представлення, а внутрішньою логікою бази даних. Для уникнення утворення однакових записів, для таблиць передбачене використання наборів атрибутів або конкретного атрибута, який однозначно визначає кожен. Подібні атрибути позначаються, як первинні ключі. Два кортежі таблиці не можуть однакові значення первинного ключа, оскільки, це призводить до порушень у ідентифікації. Реляційна база даних підтримує створення зв'язків між таблицями за допомогою зовнішніх ключів – спільних атрибутів, які відповідають в одній із таблиць первинним ключам.

Головною особливістю реляційних баз даних є виконання змін будь-якої кількості, як однієї. Такі процедури носять назву транзакції. Для використання транзакцій база даних підтримує чотири основних принципи, які забезпечують успішне виконання транзакцій:

– атомарність (Atomicity) – транзакція не буде виконана не повністю;

– узгодженість (Consistency) – до та після транзакції жодні дані в системі не мають суперечливого стану;

– ізольованість (Isolation) – жодні зміни під час виконання транзакції не будуть доступними нікому, допоки транзакція не завершена;

– довговічність (Durability) – БД гарантує, що результати транзакцій будуть збережені навіть після збоїв системи.

На відміну від попередніх моделей, база даних реляційної моделі використовує для роботи з інформацією не структуру бази даних, а її атрибути, що значно полегшує роботу з нею. Єдиним відчутним недоліком в такій концепції є деяка надмірність наповнення записів, оскільки це необхідне для утворення зв'язків між таблицями.

У результаті розгляду основних різновидів баз даних, для виконання проекту була обрана реляційна модель, як відносно просту у практичному застосуванні.

Система передбачає одну локальну базу даних, яка складається з чотирьох наборів даних – таблиць: Товар, Постачальники, Замовлення, Збут.

Атрибуті таблиці Товар, яка відображає розташований на складі товар:

– ID (унікальний код-ідентифікатор; первинний ключ);

– Model (назва моделі товару; вміщає унікальні значення);

– Type (вказує тип товару: миша, монітор, ноутбук тощо);

– Category (більш загальна категорія товару: готові комп'ютери, периферійні пристрої, додаткове обладнання, складові);

– Price (ціна за одиницю);

– Amount (кількість штук на зберіганні).

Атрибуті таблиці Постачальники:

– ID (унікальний код-ідентифікатор; первинний ключ);

– Name (назви компаній; тільки унікальні значення);

– Representative (прізвище, ім'я фізичної особи-представника);

– Supply (категорії товарі, які постачаються);

– Phone (номер телефону, за яким звертатися).

Атрибуті таблиці Замовлення, яка описує всі постачання до підприємства:

– ID (унікальний код-ідентифікатор; первинний ключ);

– DStart (дата оформлення);

– DEnd (дата запланованого виконання поставки);

– Supplier (назва компанії-постачальника);

– Model (назва моделі товару);

– Type (тип товару);

– Price (ціна за одиницю);

– Amount (ціна для продажу);

– Done (стан виконання).

Атрибуті таблиці Збут:

– ID (унікальний код-ідентифікатор; первинний ключ);

– Date (дата продажу);

– Model (назва моделі товару);

– Type (тип товару);

– Amount (кількість проданого товару за дату).

Графічне представлення структури бази даних подано на рисунку 2.7.

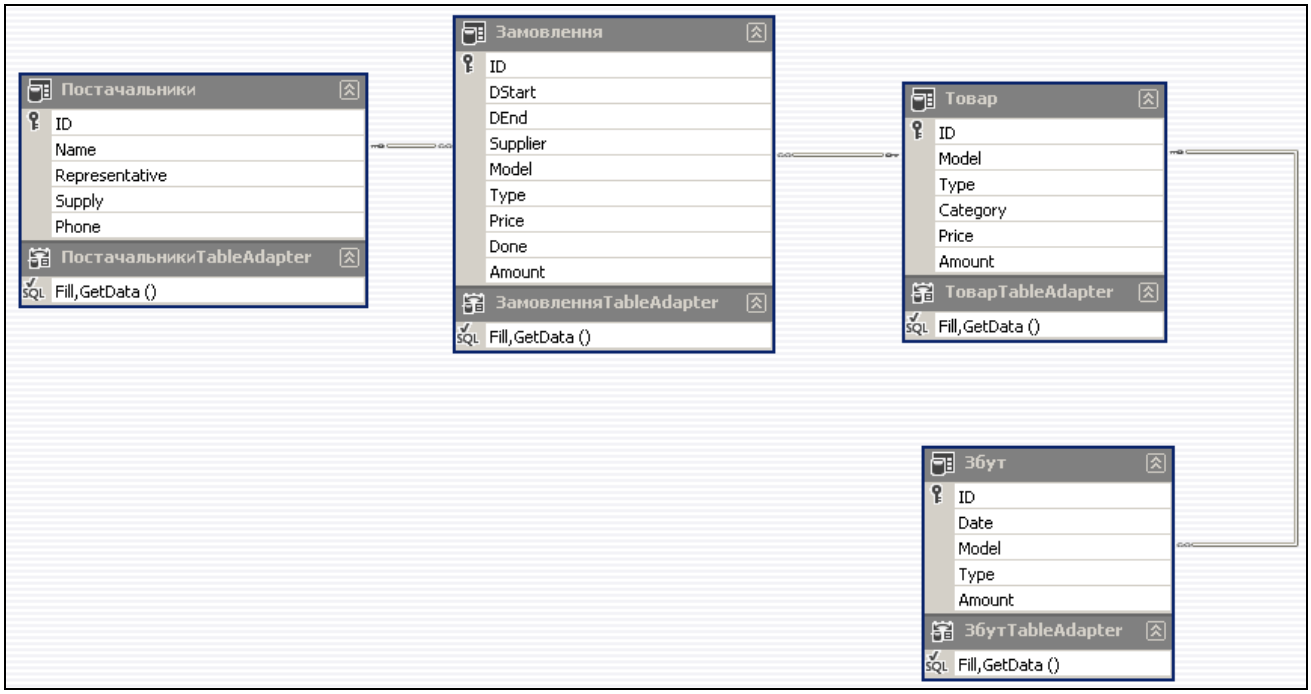


Рисунок 2.7. – Графічне представлення структури бази даних

2.3 Аналіз ти вибір технологій для реалізації програмної системи

Всі популярні мови програмування можуть бути використані для розробки програмних додатків. Здебільшого, в якості засобу зв'язку коду програми та графічного інтерфейсу користувача виступають платформи та середовища розробки, які надають необхідні бібліотеки.

В якості мови програмування обрана мова C# (Сі-Шарп). Мова програмування C# – об'єктно-орієнтована мова програмування з безпечною

системою типізації для платформи .NET. Ця мова являється доступним та досить потужним інструментом для розробки різноманітних програмних додатків.

Представляє розвиток мови C++ та Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі.

Будучи об'єктно-орієнтована мовою, в процесі написання використовує готові конструкції замість написання великої кількості коду, що значно полегшує розробку та читання коду, хоча така універсальність може негативно впливати на швидкодію додатків.

Окрім того, це – продукт компанії Microsoft, з чого можна зробити висновки щодо поширеності та актуальності мови програмування. А також про наявність великої спільноти розробників, до якої можливо звернутися у процесі праці с даною мовою програмування.

Інші немалозначні переваги мови пов'язані з далі вказаними інструментами, з якими зазвичай Сі-Шарп і застосовується.

C# працює на базі платформи Microsoft .NET. .NET Framework – це програмна платформа від компанії Microsoft, яка підтримує розробку та виконання програмних додатків різних типів: від простих додатків до веб-служб. Специфіка платформи базується на використанні Common Language Runtime (CLR) – середовищі, у якому код мов програмування перетворюється у проміжний універсальний код, який вже перетворюють у машинний.

.NET Framework складається із двох частин. Перша частина включає набір заздалегідь написаного коду (офіційно іменованого SDK, Dev Packs або «Пакети розробника»). Друга частина включає програму, яка може інтерпретувати код .NET Framework в команди для операційної системи. Ця частина, яку називають «середовищем виконання», дозволяє запускати програми, написані з використанням .NET Framework.

Таким чином, основними можливостями платформи:

– кросплатформенність – платформа здатна підтримувати більшість операційних систем лінійки Windows, підтримувати розробку під Linux та MacOS, та навіть розробку мобільних додатків;

– мультимовність – за рахунок трансформацій у проміжний код Common Intermediate Language (CIL). Розробник отримує свободу у виборі мови програмування. А, за необхідності, така структура реалізації коду дозволяє створювати проекти на кількох мовах. Найбільш популярними мовами, що підтримуються, є C#, VB.NET, C++, F#. У разі необхідності є окремі проекти, завдяки яким, стає можлива розробка під цю платформу на інших мовах;

– велика бібліотека класів та технологій – платформа підтримує величезну кількість готових бібліотек, за допомогою яких можливе виконання широкого спектру задач. Завдяки спеціальному менеджеру пакетів nuget користувач позбавляється необхідності реалізовувати логіку роботи програми на низькому рівні.

Головним недоліком відносно платформи можна вважати те, що без встановлення .NET Framework, фактично, більшість сучасних програм просто не будуть працювати на комп’ютері.

Для розробки графічного інтерфейсу буде використана технологія, представлена вище описаною – Window Forms, або скорочено WinForms. Це досить зручний і доступний інструмент, який, до того, максимально сумісний з іншими обраними засобами розробки.

Windows Forms використовується в Microsoft .NET для створення програм, для яких передбачено використання графічного інтерфейсу. Цей інструмент базується на освнові .NET Framework class library і має набагато досконалішу і зручнішу в роботі модель програмування, ніж, наприклад, програмні інтерфейси Win32 API або MFC.

По суті, Windows.Forms представляє собою комплекс керованих бібліотек, в яких описані всі дії, характерні для віконного додатку, від обміну інформацією з операційною системи комп’ютера для відслідковування взаємодій з вікном

програмного додатку, до діалогових систем, засобами формування мережевого зв’язку між пристроями тощо.

З використанням обгортки для створеного Win32 API, цей комплекс надає користувачу доступ до складових інтерфейсу Windows. При цьому, наслідуючи платформу .NET, Windows Forms не залежить від конкретної мови програмування. Тобто створення віконного додатку можливе на всіх мовах, які підтримує .NET Framework. До переваг технології також можна віднести:

– споживання малої кількості пам’яті;

– для розробки всієї логіки достатньо однієї мови програмування;

– технологія широко підтримується на операційних системах лінійки Windows;

– просте освоєння технології, використання стандартних об’єктів з використання інтуїтивно зрозумілих параметрів допомагає у використанні бібліотек;

До недоліків відносяться:

– неможливість використання додатку та технології на якійсь іншій платформі, зокрема Windows;

– для відображення додатку не використовується ресурси графічного процесора, через що спостерігається повільніша робота у порівнянні з аналогами;

– хоча технологію досі є сенс використовувати, на сьогодні вже існують більш розвинуті аналоги з більшим функціоналом.

Для розробки реляційної бази даних вибір стоїть між дуже схожими рішеннями: MySQL та Microsoft SQL Server, які були створені з різницею у лічені роки і отримали приблизно однакову популярність. Обидва мають немало як спільних рис, так і розбіжностей. У плані використання схожість роботи з базою даних та синтаксису зводить вибір між цими до звичайного суб’єктивних уподобань. Таким чином, чином вибір пав на SQL Server, як рішення, у

використанні якого мається досвід. До того ж, середовище розробки пропонує використання її полегшеної версії.

SQL Server було створено компанією Microsoft у 1987 році. На сьогоднішній день актуальною є версія 2019, яка вийшла, власне, у 2019 році. SQL Server досить довго був системою керування бази даних, розробленою виключно під Windows, однак, починаючи з версії 16, ця система доступна і на Linux.

До SQL Server виділяють наступні особливості:

– продуктивність. SQL Server працює дуже швидко;

– надійність та безпека. SQL Server надає шифрування даних;

– простота. З цієї СУБД легко працювати та вести адміністрування.

Взаємодія з базою даних базується на використанні мови SQL (Structured Query Language). Програма, яка виконує роль клієнта, використовує цю мову SQL для формування спеціальних запитів та надсилання через спеціальний програмний інтерфейс. Система керування базою даних зчитує, інтерпретує та виконує запит, після чого повертає результат виконання запиту.

Перевагами цієї системи є спеціалізація для операційної системи Windows, багатий набор інструментів розробки та керування, добре розвинений діалект SQL – Transact-SQL.

Недоліки випливають з зосередження розробки цієї системи навколо сімейства Windows, через що, якість роботи системи напряму залежить від цього операційного середовища. А можливості використання продуктів, які не належать Microsoft, досить обмежені.

В даному проекті було застосована версія Express 2016 Localdb, яка опціонально постачається разом з середовищем розробки та дещо облегшеної версією Express за рахунок функціоналу виключно для локальних баз даних.

Нарешті, в якості середовища розробки була обрана Microsoft Visual Studio 2019, у якій об’єднані всі засоби розробки, описані вище. Visual Studio – це лінійка продуктів, започаткована наприкінці минулого століття, як засіб полегшення роботи з мовами, на кшталт C++. Згодом продукт набув

функціоналу та поширення і зараз розповсюджується у різних версіях для різних типів користувачів (хоча фактична різниця між версіями мінімальна).

Microsoft Visual Studio – це середовище програмування, розроблене компанією Microsoft. MVS дозволяє створювати кросплатформені проекти різними мовами програмування, такими як Visual Basic, Visual C#, Visual C++, Visual F# та інші. У Visual Studio розробник може проводити розробку веб-сайтів, веб-служб, писати консольні програми, а також програми з графічним інтерфейсом. Інтерфейс середовища розробки представлено на рисунку 2.8.

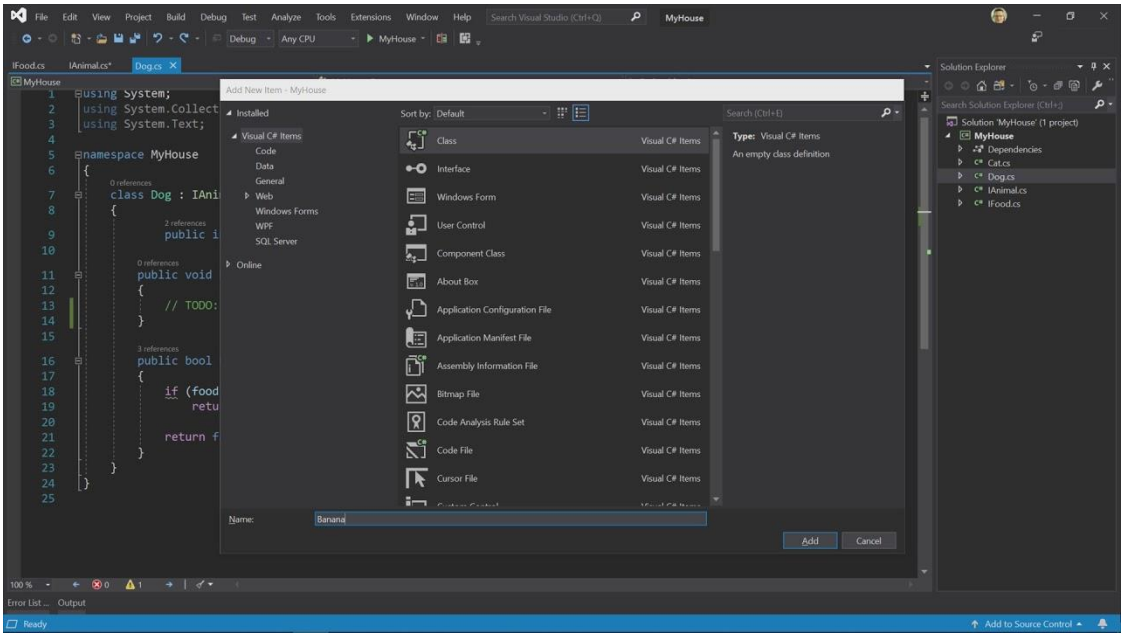


Рисунок 2.8. – Зразок інтерфейсу Microsoft Visual Studio

Також MVS підтримує різноманітні доповнення. Наприклад, згаданий до платформи .NET Framework менеджер пакетів nuget стає доступний через інтерфейс середовища розробки, завдяки чому стає можливе доповнення стандартних бібліотек новими, необхідними для конкретного проекту.

Як і будь-яке середовище розробки, до складу Visual Studio входять:

– редактор коду, який здатен підсвічувати синтаксис, помилки, типи даних та інше;

– компілятор та інтерпретатор для перетворення мов програмування у машинний код. Visual Studio використовує обидва в залежності від проекту;

– засоби автоматизації складання;

– дебаггер, або відладник для дослідження коду на предмет помилок

Перевагами середовища розробки MVS вважаються:

– підтримка великої кількості мов програмування;

– вбудований веб-сервер, що полегшує розробку веб-додатків;

– автоматичне додавання коду до стандартних елементів керування;

– автоматичне форматування коду під час його написання (підсвічування, вставка відступів);

– велика спільнота, до якої можна звернутися по допомогу;

– програма-інсталятор, який надає можливість встановлення, окрім самого середовища розробки, додаткового програмного забезпечення, необхідного для роботи.

До значних недоліків виділяють MVS відносять:

– масивність – середовище розробки разом з додатковим програмним забезпеченням та інструментарієм, навіть для однієї області розробки, займає немало місця, особливо на системному диску, через що можуть виникнути проблеми з працездатністю робочої машини;

– обмеженість офіційними робочими платформами Windows та Mac.

2.4 Проектування інтерфейсу користувача

Виходячи з завдання проекту, інтерфейс розроблюваного програмного забезпечення буде розроблюватися з нуля засобами Windows Forms.

Інтерфейс розроблюваної системи обліку матеріальних цінностей утворено чотирма формами. Перша форма призначена для авторизації користувача та для отримання доступу до основної програми. До елементів форми, яким надано функціонал, відносяться два текстових контейнери для

логіну та паролю, і кнопка введення. Текст в контейнері для пароля приховується. Рух форми по екрану заблоковано. Зображення форми представлено на рисунку 2.9.

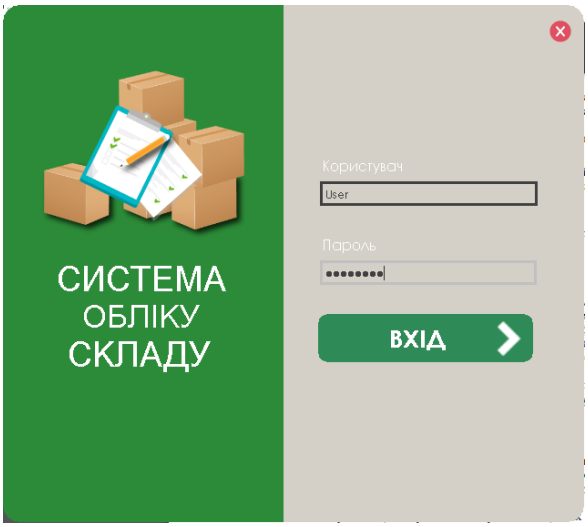


Рисунок 2.9. – Інтерфейс форми авторизації користувача

Основна програма розділена на дві основні (Product, Suppliers) та одну додаткову форму Orders. Основні форми мають відмінності за функціональним призначенням, але графічне оформлення інтерфейсу фактично однакове.

Вікно основної форми можна розділити на три великі частини (рисунок 2.10., рисунок 2.11. і рисунок 2.12.):

– бокова панель кнопок;

– робоча площа панелей;

– нефункціональна основа.

Бокова панель складається з кнопки яка відображає поточну категорію «Товар» записів бази даних, натиск на яку, змінює категорію на «Постачальники», відкриває другу основну форму замість поточної. Нижче йдуть кнопки, які визначають режим роботи з таблицею: додавання нового запису, оновлення та видалення існуючого. Далі йдуть кнопки, які звертаються до таблиць з виконаними та невиконаними постачаннями товару, та кнопка звернення до таблиці збуту товару. Натискання на кнопку режиму, впливає на

структуру робочої площі панелей та виділяється додатковим графічним елементом.

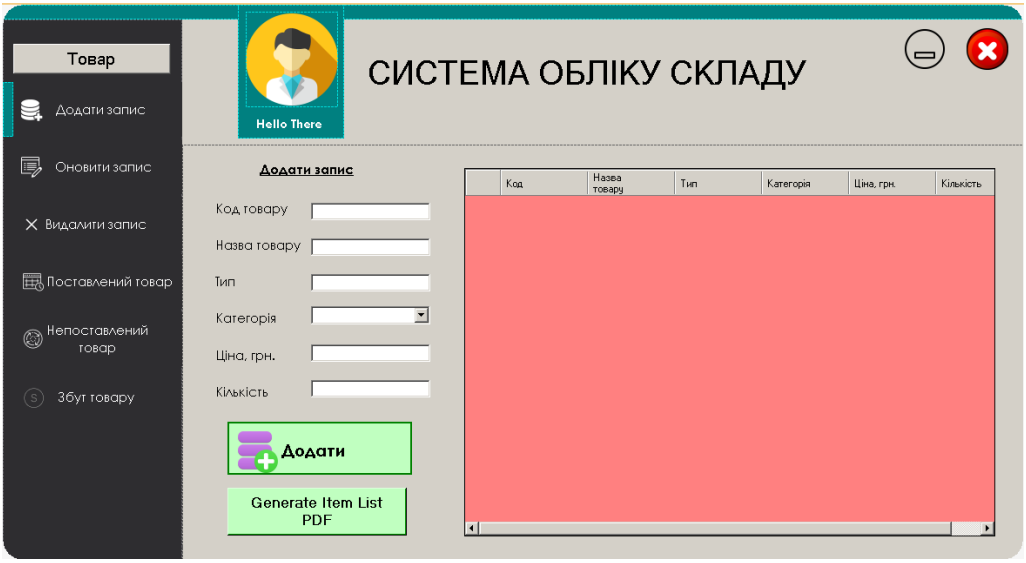


Рисунок 2.10. – Основна форма розроблюваного програмного забезпечення в режим додавання запису

Робоча площа панелей на рисунку 2.10. відокремлена від нефункціональної частини вікна лінією зверху, яка проходить над контейнером в правій частині. Вона представлена набором панелей з елементами: текстові рядки, текстові контейнери, контейнер комбінованого списку, кнопки взаємодії з таблицею. Також на цих панелях розташовано контейнер для відображення наборів даних, в якій буде відображатися вміст бази даних. Для коректної роботи програми, в це вікно вручну додано стовпчики з заголовками українською мовою, які відповідають атрибутам таблиці бази даних. Текстові рядки та відповідні їм текстові контейнери відповідають атрибутам баз даних.

В залежності від режиму роботи з базою даних, обраною кнопкою бокової панелі, відображається панель з кнопкою відправлення відповідного запиту та кнопка створення PDF-файлу.

При виборі режиму з таблицями постачань з’являються панелі з кнопками для зміни статусу постачання, його відміни (видалення) та створення PDF-

файлу. На панелі з невиконаними постачаннями розташована додаткова кнопка для створення та оновлення записів, яка викликає нову форму (рисунок 2.11.).

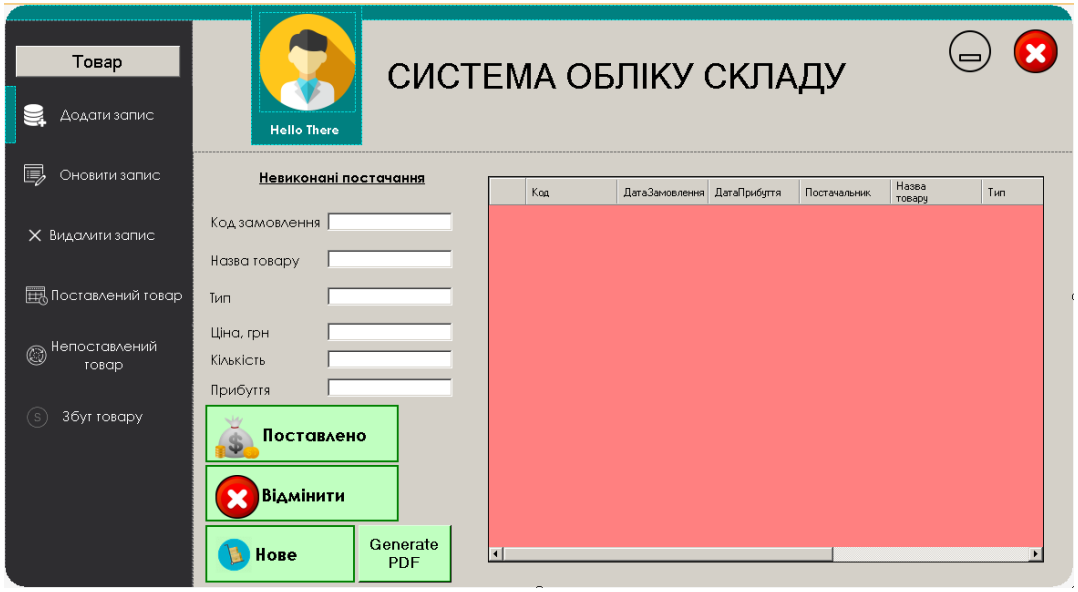


Рисунок 2.11. – Основна форма розроблюваного програмного забезпечення в режим відображення невставленого товару

Панель для таблиці збуту відрізняється розміщенням кнопок для видів редагування (рисунок 2.12.).

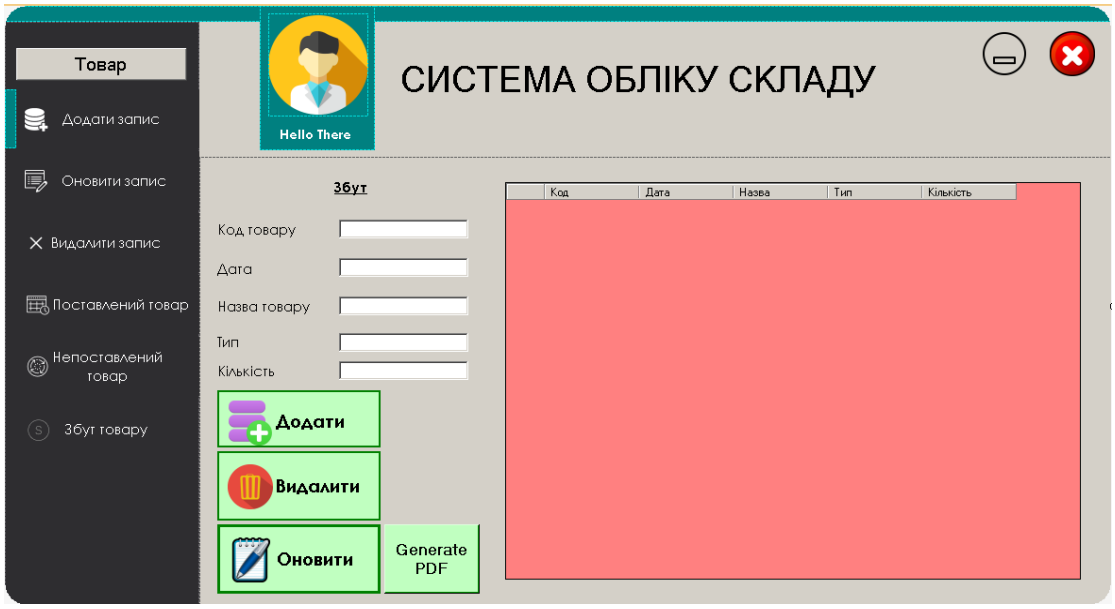


Рисунок 2.12. – Основна форма розроблюваного програмного забезпечення в режим відображення збуту товару

Друга основна форма для режиму таблиці «Постачальники» має аналогічну структуру з урахуванням зміни структури таблиць. Панелі для таблиць постачань та збуту мають однакову структуру (рисунок 2.13.).

При переключені режимів основної форми поточна форма приховується і відкривається друга.

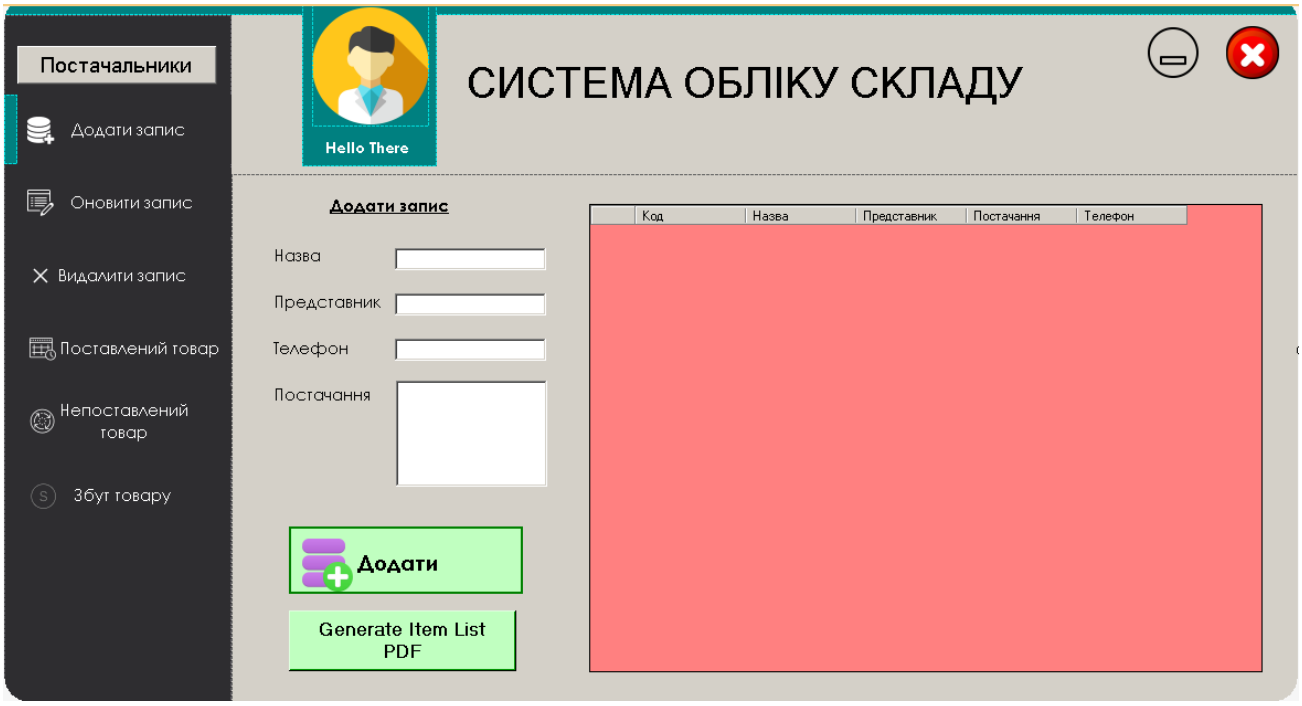


Рисунок 2.13. – Основна форма в режимі «Постачальники»

Додаткова форма для створення та оновлення записів про замовлення постачань має структуру з аналогічною логікою, що й робочі панелі основних форм, тільки в даній формі не відбувається жодних переключень, а тільки розташовані кнопки для надсилання запитів у базу даних та контейнер для масиву даних.

Графічне представлення форми подано на рисунку 2.14.

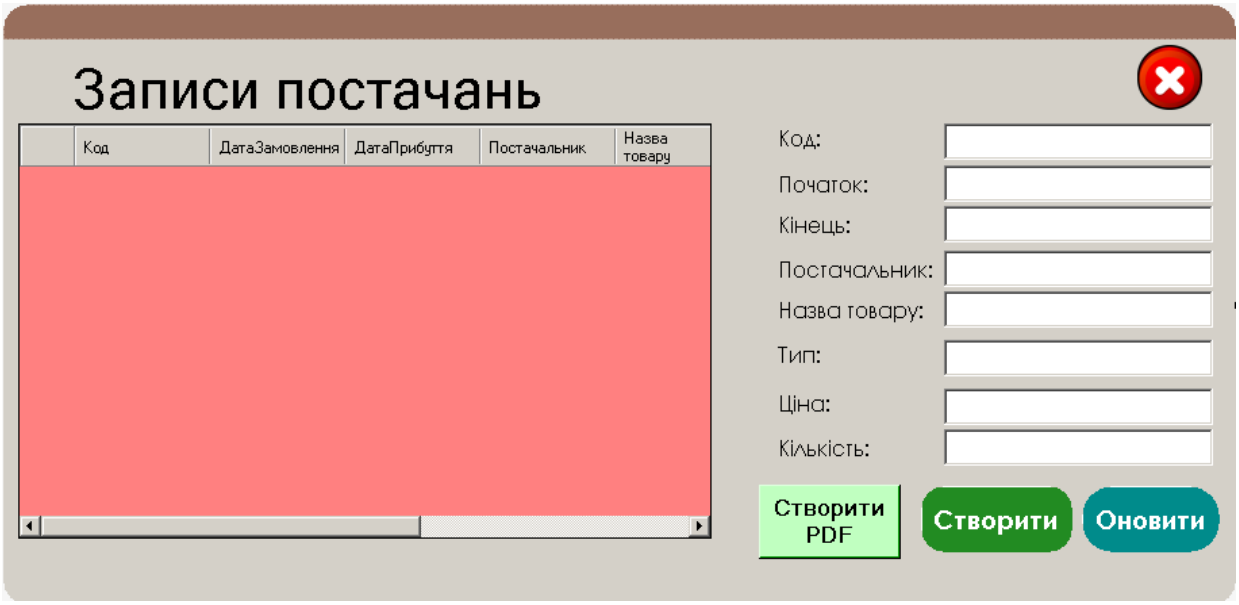


Рисунок 2.14. – форма додавання та оновлення нових записів замовлень постачань

В даному розділі було проведено дослідження існуючих архітектур систем, які працюють з базами даних. У результаті дослідження була сформована архітектура розроблюваного проектного додатку з урахування його специфіки. Були виділені основні елементи системи та визначена специфіка взаємодії між ними. Проведено огляд основних видів баз даних за різновидами моделей даних. В результаті прийнято рішення обрання реляційної бази даних на основі використання Microsoft SQL Server. Було спроектовано інтерфейс, визначені його функціональні елементи та логіка використання форм. Для розробки проектного програмного забезпечення були обрані мова програмування, платформа розробки, засоби створення графічного інтерфейсу користувача та середовище розробки.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Детальне проектування модулів

Проводячи проектування модулів програмної системи, будуть описані способи роботи модулів і способи взаємодії між собою. До опису буде подана: загальна структура, особливості архітектури та способи взаємодії між модулями програми. Також проведено формування діаграми класів. Розпочнемо з опису модулів та їх взаємодії між собою.

У попередньому розділі були розглянуті функції розроблюваного програмного забезпечення, а також структура графічного інтерфейсу і функціональна логіка його елементів. В результаті можна виділити наступні модулі:

- запуск програми;
- авторизація;
- виведення даних з бази даних;
- вибір режиму роботи з базою даних;
- редагування бази даних;
- створення PDF-файлу.

Далі буде продемонстрована структуру даних всередині розроблюваного програмного забезпечення засобами діаграми класів. Слід зазначити, як згадувалось вище, форми, розроблені засобами мови програмування C# та технологією Windows Forms представляють собою екземпляри класу Form. На діаграмі класів вони будуть відображені, як самостійні класи, оскільки містять визначення функцій, які відбуваються унаслідок реакцій на взаємодію користувача з елементами графічного інтерфейсу розроблюваного програмного забезпечення, тобто містять опис роботи модулів. Утворена діаграма класів представлена на рисунку 2.15.

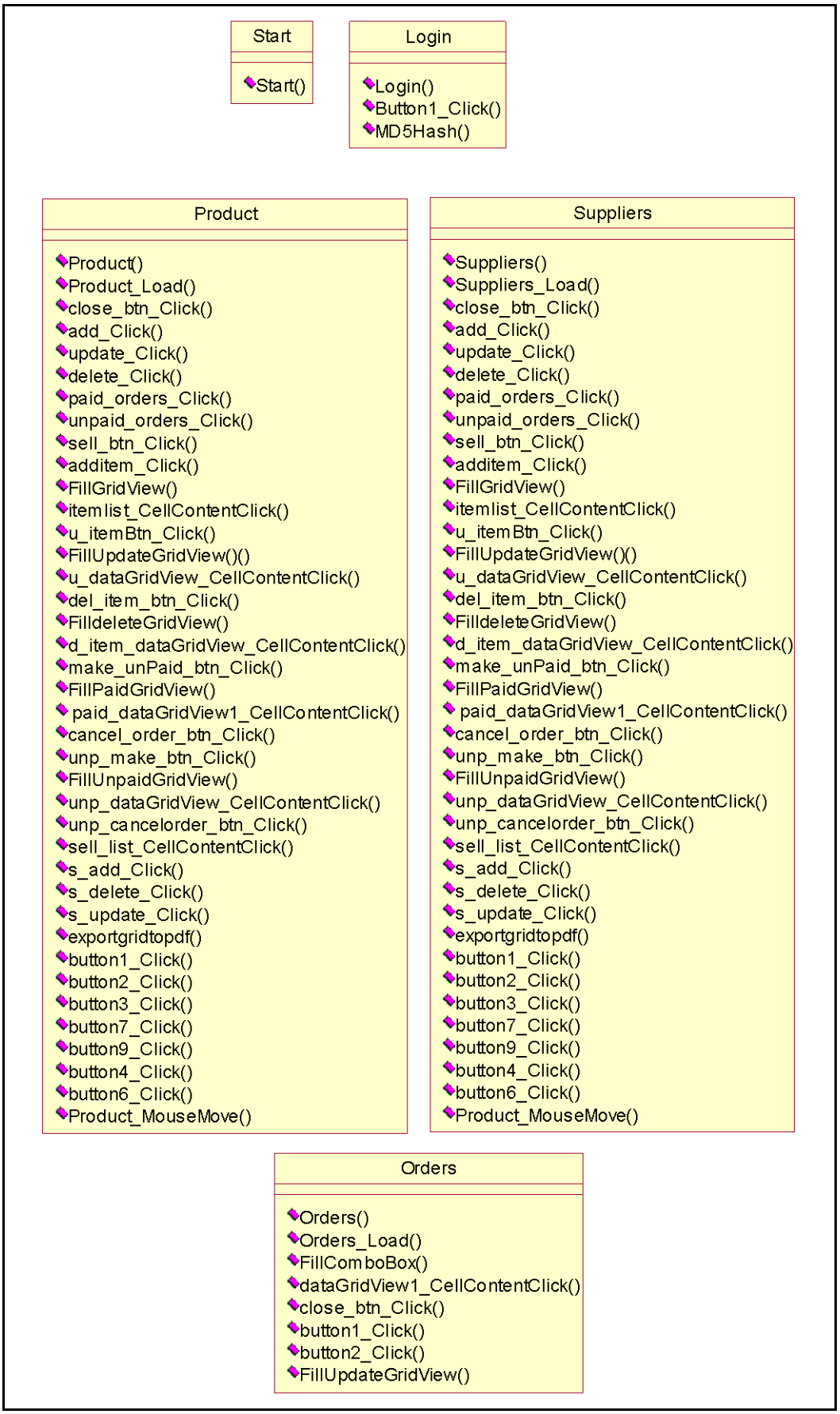


Рисунок 3.1. – Форма додавання та оновлення нових записів замовлень постачань

3.2 Реалізація логіки додатка

Як вже була частково розкрита логіка роботи розроблюваного програмного забезпечення при описі інтерфейсу користувача, робота програми починається зі стандартної ініціалізації програми, яка відкриває форму авторизації.

Форма очікує введення двох змінних, після чого натиснення кнопки введення даних викликає обробку натискання, в якій вхідні данні перетворюються на змінні, одночасно з цим проходячи процес перетворення у хеш-функції.

```
 private void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                string user = MD5Hash(Textbox1.Text.Trim());
                string user1 = "8f9bfe9d1345237cb3b2b205864da075";
                string password = MD5Hash(Textbox2.Text.Trim());
                string password1 = "5f4dcc3b5aa765d61d8327deb882cf99";
                if (password == password1 && user == user1)
                {
                    Product product = new Product();
                    this.Hide();
                    product.Show();
                }

                else
                {
                    MessageBox.Show("Неправильний пароль та/або
користувач");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
```

Нижче приведено алгоритм самого процесу хешування вхідних значень. Отримуючи на вході рядок, утворюється змінний рядок, масив байтів, у який буде перетворений вхідний рядок та об’єкт класу, що виконує алгоритм обчислення хешу на основі попереднього масиву. На виході функція видає

змінний рядок, утворений з байтів, перетворених у шістнадцятирічний рядок довжиною в два символи.

```
        public static string MD5Hash(string input)
        {
            StringBuilder hash = new StringBuilder();
            MD5CryptoServiceProvider md5provider = new
MD5CryptoServiceProvider();
            byte[] bytes = md5provider.ComputeHash(new
UTF8Encoding().GetBytes(input));

            for (int i = 0; i < bytes.Length; i++)
            {
                hash.Append(bytes[i].ToString("x2"));
            }
            return hash.ToString();
        }
```

Логіка програмного забезпечення надає основні типи взаємодії з базою даних: перегляд, додавання, видалення, оновлення. По аналогії з описом графічного інтерфейсу, основу логіки додатку розглянемо на реалізації функцій для форми Product. При ініціалізації форми відбувається автоматичний вибір режиму додавання запису, однак, оскільки всі функціональні панелі та їх таблиці відразу завантажуються, але приховуються, окрім обраної, код відразу виконує їх налаштування.

```
  void FillGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
Application.StartupPath + @"\MyBD.mdf;Integrated Security=True;Connect
Timeout=30");
            itemlist.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Товар", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                itemlist.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),dr[4].ToString(), dr[5].ToString());
            }
            dr.Close();
            conn.Close();
        }
```

Даний фрагмент демонструє виконання перегляду змісту бази даних. Він заповнює таблицю режиму створення запису про товар на складі кортежами з відповідної таблиці бази даних. Для формування SQL-запиту відбувається підключення до бази, потім створюється SQL-запит для перегляду змісту таблиці. Поки SQL-запит зчитує дані з бази даних, рядки у контейнері для масиву даних заповнюються постовпчиково через створений об’єкт класу зчитування. По завершенню виконання запиту підключення до бази даних перериваються. Аналогічна функція записана для вибору кожного режиму роботи з базою даних.

Наступний фрагмент дозволяє додавання нового запису. Для цього контейнери активної панелі заповнюються даними. Після ініціалізації створення запиту, спочатку перевіряється наявність всіх введених даних. За цим слідує створення підключення бази даних та створення запиту на утворення кортежу з атрибутів, які сформовані з відповідних записів текстових контейнерів. Після виконання запиту підключення переривається, а контейнери для вводу очищуються. У форму виводиться оновлена таблиця. Як у разі успіху, так і виникнення помилки, інтерфейс виводить відповідне повідомлення.

```
private void additem_Click(object sender, EventArgs e)
        {
            if(model.Text!="" && part.Text!="" && comboBox1.Text!="" &&
price.Text!="" && instock.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
Application.StartupPath + @"\MyBD.mdf;Integrated Security=True;Connect
Timeout=30");
                    string query = "INSERT INTO Товар(Model, Type,
Category, Price, Ammount) VALUES(@Model, @Type, @Category, @Price,
@Ammount)";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Model",
model.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
part.Text.ToString());
                    cmd.Parameters.AddWithValue("@Category",
comboBox1.Text.ToString());
```

```
                    cmd.Parameters.AddWithValue("@Price",
price.Text.ToString());
                    cmd.Parameters.AddWithValue("@Ammount",
instock.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис додано!");
                    model.Clear();
                    part.Clear();
                    price.Clear();
                    instock.Clear();
                    comboBox1.SelectedIndex = -1;
                    FillGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповність всі поля");
            }
        }
```

Оновлення запису здійснюється по аналогічній схемі. Однак, перед цим треба вибрати цільовий запис, натисканням на рядок таблиці в контейнері для масивів, що також передбачено кодом. Це автоматично заповнить текстові контейнери існуючими даними, які можна змінити. Після цього, виконується підключення та формування відповідного запиту, оновлюється запис, перевивається підключення та виводиться оновлена таблиця. Як у разі успіху, так і виникнення помилки, інтерфейс виводить відповідне повідомлення.

```
private void u_itemBtn_Click(object sender, EventArgs e)
        {
            if (u_modelTxt.Text != "" && u_partTxt.Text != "" &&
u_typeCombo.Text != "" && u_priceTxt.Text != "" && u_stockTxt.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
Application.StartupPath + @"\MyBD.mdf;Integrated Security=True;Connect
Timeout=30");
                    string query = "update Товар set Model = @Model,Type
= @Type,Category = @Category,Price =@Price, Ammount = @Ammount where ID =
'"+ u_itemcodeTxt.Text +"' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
```

```
                    double Price = double.Parse(u_priceTxt.Text);
                    cmd.Parameters.AddWithValue("@Model",
u_modelTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
u_partTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Category",
u_typeCombo.Text.ToString());
                    cmd.Parameters.AddWithValue("@Price", Price);
                    cmd.Parameters.AddWithValue("@Ammount",
u_stockTxt.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис оновлено");
                    u_itemcodeTxt.Clear();
                    u_itemcodeTxt.Enabled = false;
                    u_itemcodeTxt.Text = "Id Auto Number";
                    u_modelTxt.Clear();
                    u_partTxt.Clear();
                    u_priceTxt.Clear();
                    u_stockTxt.Clear();
                    u_typeCombo.SelectedIndex = -1;
                    FillUpdateGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть запис");
            }
```

Аналогічним чином виконується видалення запиту, тільки необхідно просто вибрати запис та ініціалізувати створення запиту.

```
private void del_item_btn_Click(object sender, EventArgs e)
        {
            if (d_modelTxt.Text != "" && d_partTxt.Text != "" &&
d_typeCombo.Text != "" && d_priceTxt.Text != "" && d_instockTxt.Text !=
"")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +
Application.StartupPath + @"\MyBD.mdf;Integrated Security=True;Connect
Timeout=30");
                    string query = "delete from Товар where ID = '" +
d_itemcodeTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
```

```
                    conn.Close();
                    MessageBox.Show("The item has been successfully
deleted!");
                    d_itemcodeTxt.Clear();
                    d_itemcodeTxt.Enabled = false;
                    d_itemcodeTxt.Text = "Id Auto Number";
                    d_modelTxt.Clear();
                    d_partTxt.Clear();
                    d_priceTxt.Clear();
                    d_instockTxt.Clear();
                    d_typeCombo.SelectedIndex = -1;
                    FilldeleteGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }
```

При роботі з таблицями збуту структура функцій аналогічна. При режимі роботи з поставленими та непоставленими товарами видалення аналогічне, але для панелі кнопка ініціює специфічну зміну атрибуту статусу здійснення постачання, завдяки якому записи однієї таблиці відображаються у різних таблицях цього режиму.

Повноцінне додавання та оновлення будь-якого запису можливе через виклик додаткової форми, в якій ці функції реалізовано так само, як описано вище.

Весь наданий опис реалізації функцій та логіки функціоналу програми відповідає таким у другій основній формі.

Весь код виконання функцій та реакцій вікна (лістинг) програми подано у додатку В.

3.3 Керівництво користувача

Кожне підключення програми супроводжується авторизацією користувача (рисунок 3.2.). Вводиться логін та пароль.

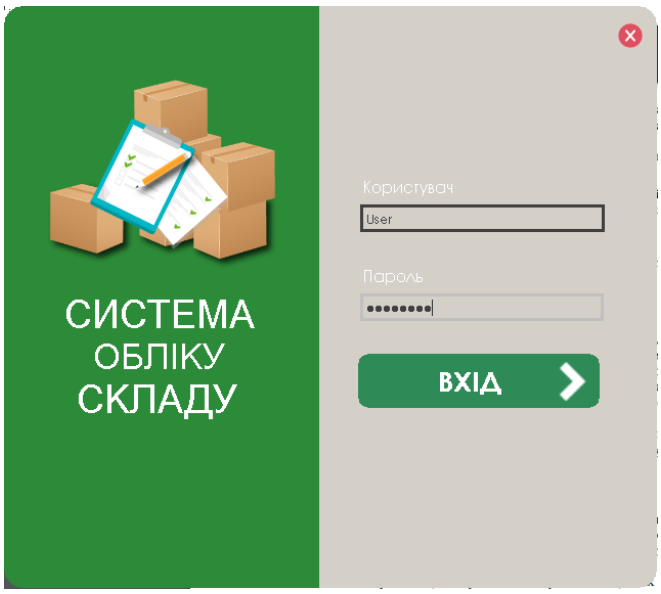


Рисунок 3.2. – Вікно авторизації

У разі проходження перевірки, вікно зникає і відкривається основне робоче вікно програми (рисунок 3.3.).

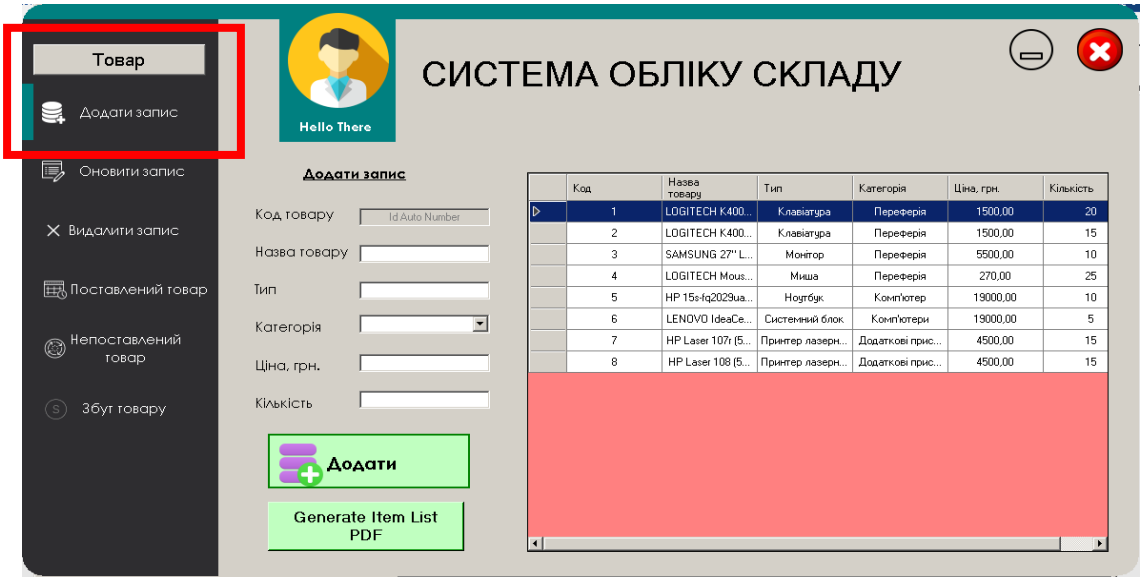


Рисунок 3.3. – Вікно основної програми.

В основному вікні програма відразу вибирається режим додавання запису – що відображено виділенням кнопки в лівій частині вікна – для таблиці з товарами, що буде видно по напису на великій кнопці з написом «Товар». На рисунку 3.3. червоною рамкою виділено обраний режим роботи з записом та назва таблиці, з якою працює цей режим. В робочій частині вікна, яка займає більшу частину, відразу з'явиться таблиця по товарам на складі. Записи таблиці товарів містять записи про код товару, назву моделі, тип, загальну категорію, ціну за одиницю та кількість в наявності.

Перехід між режимами роботи з таблицями здійснюється за допомогою натискання миші на відповідній кнопці в лівій частині. У випадку режимів редагування таблиці Товари при зміні режиму у робочій області буде з'являтися відповідна кнопка для здійснення редагування (рисунок 3.4.).

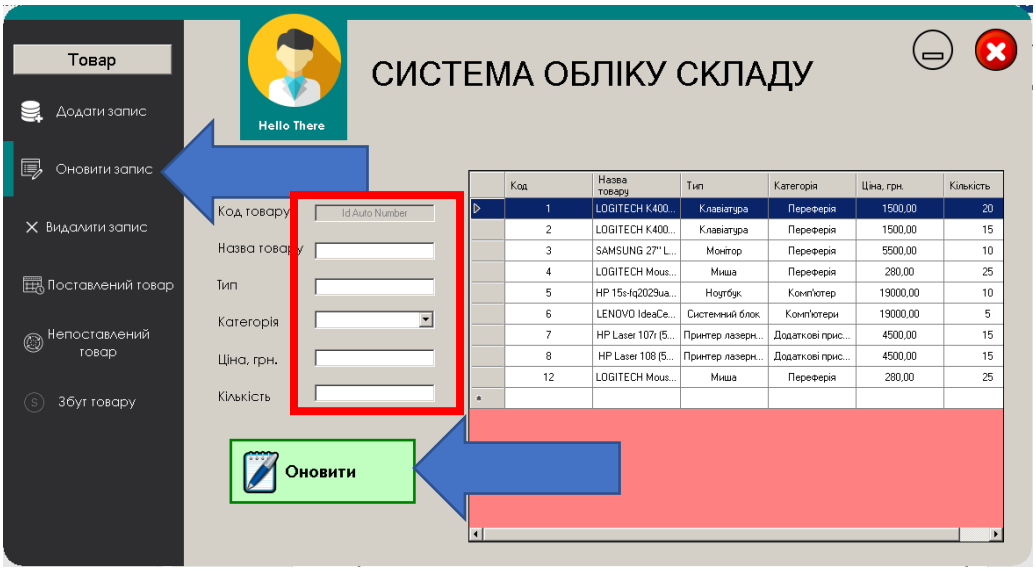


Рисунок 3.4. – При виборі режиму оновлення запису з'являється відповідна кнопка

Для здійснення операцій з записами таблиці необхідно вибрати потрібний режим та діяти відповідно нього: при додаванні достатньо ввести дані у спеціальні комірках (на рисунку 3.4 виділено червоним) та натиснути відповідну кнопку. При оновлені треба обрати цільовий запис, після чого комірки автоматично заповняться даними, які можна змінити та зберегти зміни

через відповідну кнопку. Для видалення достатньо тільки обрати запис та натиснути відповідну кнопку. В режимі додавання запису також можна автоматично заповнити комірки вибором запису.

У режимі відображення збуту товару таблиця змінюється, а функціональні кнопки додавання зміни та видалення разом з'являються в робочій області. Їх використання аналогічне тому, що описане вище. Записи таблиці Збуту містить атрибути: код, дата продажу, назва моделі, тип товару та кількість.

У режимі «Поставлений товар» можна видаляти запис, або позначати, як неприбувший, що перенесе його у таблиці режиму «Непоставлений товар». В цьому режимі дії аналогічні, але також додається кнопка «Нове», яка відкриває додаткове вікно для повноцінного додавання записів та їх змін (рисунок 3.5.). Записи таблиці замовлень містить атрибути: код, дата замовлення, планова дата здійснення постачання, назва постачальника, назва моделі, тип товару, ціна за одиницю, кількість та стан виконання замовлення.

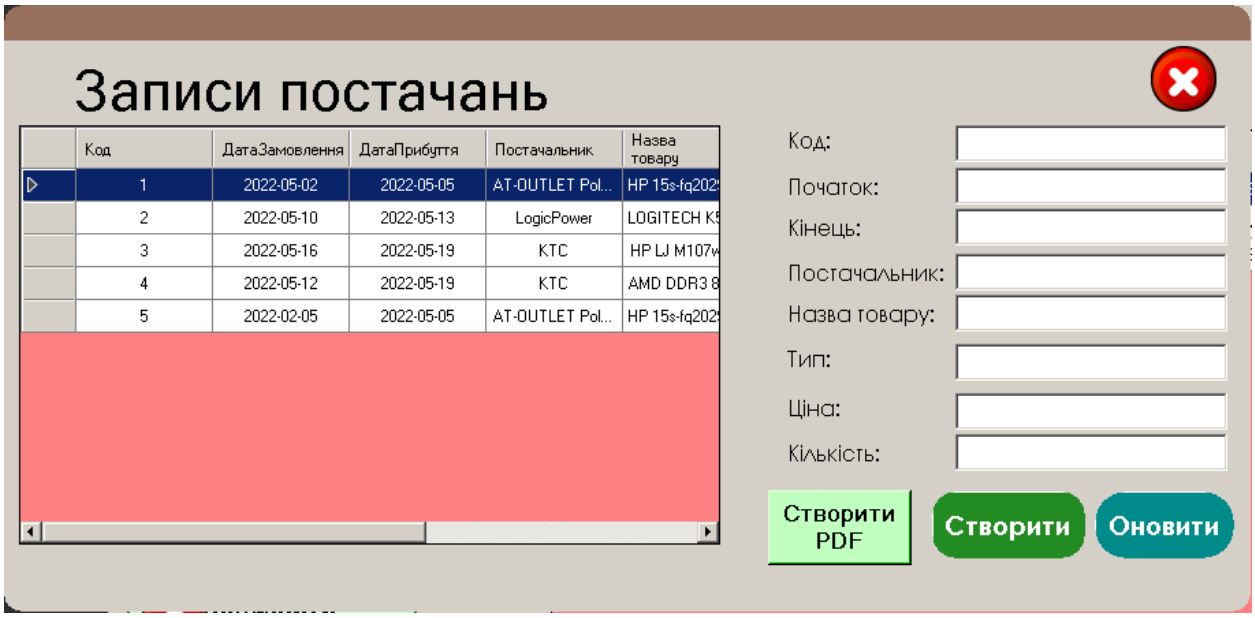


Рисунок 3.5. – Вікно створення та оновлення записів про замовлення

За допомогою самої верхньої кнопки в лівій частині вікна можна змінити таблицю з записами, якої буде працювати програма. Принцип роботи такий самий, як і описаний вище.

В програмі можливе створення файлу форма PDF, в якому буде таблиця з записами з таблиці, яка зараз відображає програма. Для цього необхідно натиснути відповідну кнопку в режимі додавання запису, поставлених і непоставлених замовлень, та у вікні додавання та оновлення записів щодо замовлень.

В усіх програмних вікнах можна закрити програму через класичну кнопку з хрестиком. А також можна згорнути основні вікна через спеціальну кнопку біля кнопки закриття. Така кнопка відсутня у додатковому вікні та вікні авторизації.

3.4 Вимоги до технічних та програмних засобів

Для використання програмного забезпечення необхідно:

- операційна система Windows;
- .NET Framework версії 4.6.1 або новіше;
- Microsoft SQL Server Express 2016 або новіше.

З урахуванням додаткового програмного забезпеченням, мінімальні системні вимоги наступні:

- 64-розрядний процесор з частотою 1 ГГц;
- 1 Гб оперативної пам’яті;
- 10 Гб дискового простору;
- периферійні пристрої: монітор, клавіатура, комп’ютерна миша.

3.5 Розгортання та встановлення системи

За умови наявності додаткового програмного забезпечення, вказаного у вимогах, достатньо копіювання файлів системи на дисковий простір

комп’ютера. Перед запуском треба достовіритися, що у папці з виконавчим файлом розташовані файл бази даних та шрифту Arial.ttf.

В даному розділі було проведено проектування модулів розроблюваного програмного забезпечення, визначена їх структура та взаємодії між ними. Також відбулася реалізація логіки додатку з використанням мови програмування, фрагменти коду якої описали виконання основних функцій. Сформувано керівництво користувача, яке описує роботу з програмним додатком. Були визначені вимоги до технічних та програмних засобів для запуску системи, а також описаний процес розгортання та встановлення системи.

4. ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Вибір та обґрунтування методів тестування додатку

Тестування програмного забезпечення – це метод визначення відповідності фактичного результату роботи розроблюваного програмного забезпечення передбаченим вимогам, що дозволяє отримати інформацію щодо того, чи наявні в програмі дефекти. Складається з використанням ручних або автоматизованих інструментів, дотримуючись завчасно встановлених алгоритмів для оцінювання одного або декількох критеріїв програми.

Загальна мета тестування – знаходження дефектів програмного продукту, що входить до цілей забезпечення якості в межах стандартного процесу розробки програмного забезпечення. Однак тестування охоплює не тільки виявлення дефектів, але й організацію коригування програми у випадку знаходження недоліків. Таким чином, можна сформувати такі цілі тестування програмного забезпечення:

– знаходження дефектів на усіх етапах життєвого циклу додатку;

– перевірка того, чи був виявлений дефект успішно усунутий;

– дослідження на предмет знаходження нових дефектів в системі, які могли виникнути у результаті виправлення вже визначених.

Виходячи з цілей, перед тестуванням ставляться такі завдання:

– знаходження дефектів моделі системи;

– знаходження дефектів кодування;

– знаходження помилок і недоліків взаємодії системи з оточенням, а також зовнішніми компонентами і системами;

– знаходження дефектів інтеграції програмного забезпечення;

– знаходження недоліків продуктивності системи;

– перевірка програмного продукту на стійкість до перевантажень;

– перевірка реакції програмного продукту на використання помилкових вхідних даних і відмови при збільшенні обсягів даних, що оброблюються;

– дослідження програми на наявність недоліків у системі безпеки, можливостей несанкціонованого доступу до конфіденційних даних;

– контроль виправлення виявлених у процесі тестування дефектів;

– виявлення регресії системи в процесі розробки.

Програмне забезпечення характеризується постійним супроводом тестування протягом всього життєвого циклу, починаючи від проектування і закінчуючи невизначено довгим етапом експлуатації, та безпосередньо пов’язане з управлінням вимогами і змінами, адже метою тестування якраз є можливість переконатися у відповідності програм заявленим вимогам.

Робота програмного забезпечення вважається правильною, якщо вона відповідає наступним вимогам:

– при наданні коректних даних, програма формує правильну відповідь;

– відхилення некоректних вхідних даних;

– відсутність «зависань» та неочікуваного припинення роботи програмного забезпечення незалежно від вхідних даних;

– функціонування програми не обмежено часовими рамками;

– програма працює без збоїв і виконує всі необхідні функції в повному обсязі.

Переваги тестування програмного забезпечення:

– своєчасно організоване тестування дозволить зменшити витрати на програмний продукт, пов’язані з усуненням дефектів;

– тестування нівелює ризики витоку інформації;

– головна мета будь-якого проекту принести свою користь користувачам;

– тестування забезпечує дотримання програмного продукту раніше визначеним специфікаціям, бізнес-вимогам та вимогам до функціоналу;

– UI/UX тестування забезпечує кращий користувацький досвід, що сприяє вибору користувачем програмного продукту, коли при наявності кількох

подібних рішень, головним критерієм стає зручність у користуванні;

– як наслідок, правильне проведення користувача сприяє формуванню довіри користувачів розробнику.

Тестування проводиться за методиками «ящиків», які описують рівень взаємодії особи-тестувальника з внутрішньою будовою об’єкта тестування. Таким чином виділяють тестування «чорного ящика», коли проведення тестів відбувається через використання виключно інтерфейсу програми без прямої взаємодії з її внутрішніми складовими, з якими напряму працюють по методології «білого ящика».

Різноманіття підходів до процесу тестування формує різноманітну класифікацію. Зазвичай тестування програмного забезпечення розділяють на три категорії:

– функціональне тестування;

– нефункціональне тестування або тестування продуктивності;

Функціональне тестування характеризується дослідженням функціональних характеристик додатка та перевіркою реальної поведінки реалізованого функціоналу на дотримання відповідності згідно специфікацій та бізнес-вимог. Фактично, процес функціонального тестування передбачає імітацію реального використання цільового програмного забезпечення.

Нефункціональне тестування досліджує програму на предмет того, «як» вона працює. До цього тестування відносять тести пов’язані зі швидкодією роботи програми та зручністю взаємодії з її інтерфейсом.

У функціональному тестуванні виділяють модульне, інтеграційне, системне та приймальне тестування.

Модульне тестування (воно ж Unit-тестування) включає в себе тестування окремих найменших функціональних елементів розроблюваного програмного забезпечення. Метою такого тестування є переконання коректності виконання фрагментом коду свого призначення. Зазвичай модульне тестування проводиться паралельно з написання коду для окремої функції.

Інтеграційне тестування передбачає перевірку взаємодії окремих об’єктів системи, як єдиної групи. Метою тестування є визначення помилок, які можуть виникнути при взаємодії між модулями.

Системне тестування представляє собою повне тестування повністю сформованого програмного продукту по методології «чорного ящика». Таке тестування досліджує відповідність функціоналу програмного додатку до вимог зазначених на ранніх етапах розробки.

Приймальне тестування частково повторює собою системне тестування. Але, якщо в системному тестуванні участь приймають розробники програмного забезпечення, в даному випадку тестування проводиться кінцевим користувачем за його баченням критеріїв. Таке тестування дає змогу зрозуміти, чи готова програма до використання реальними користувачами.

У процесі виконання проекту буде описано функціональне тестування, як засіб дослідження основних елементів розроблюваного програмного забезпечення на здатність виконувати своє призначення.

Інтегроване середовище розробки Microsoft Visual Studio 2019 надає змогу автоматизації тестів на базі платформ MSTest, NUnit та xUnit.

MSTest – вбудована в середовище розробки платформа від Microsoft, яка підтримує технології, на яких базуються програми від цього виробника.

NUnit – платформа з мови Java на C# для отримання можливостей взаємодії з платформами сімейства .NET. xUnit – це перероблена версія цієї платформи з розширеним функціоналом.

Усі ці платформи так чи інакше надають інструменти для проведення функціонального тестування, тому вибір для даного програмного додатку досить вільний. Специфікою тестування під час виконання даного проекту є те, що процес тестування виконувався вручну за допомогою виділення фрагментів коду та використання тимчасових графічних форм, без застосування засобів автоматизації.

4.2 Розробка тестових сценаріїв

Модульне тестування представлене в виді виконання окремих функцій, записаних у код конкретного модуля, який в межах розробленої програми представлені кодом оброки взаємодії з формами графічного інтерфейсу користувача. Нижче приведено таблицю, яка описує тестові сценарії і сформована за наступними категоріями: назва сценарію тестування, назва тестованого модуля (в даному випадку – форми інтерфейсу), назва функціонального методу, набір вхідних даних для обробки методом, очікуваний результат після виконання функції метода.

Для функціонального тестування додатку використовувався стаціонарний персональний комп'ютер на операційній системі Windows 7.

В таблиці 4.1 приведена частина тестових сценаріїв модульного тестування.

Таблиця 4.1 – Тестові сценарії функцій додатку

Назва сценарію	Модуль	Метод	Вхідні дані	Очікуваний результат
1	2	3	4	5
Авторизація користувача	Login	Button1_Click	Логін, пароль	Поява основного вінка програми
Хешування введених даних	Login	MD5Hash	Текс User	**8f9bfe9d1345237cb3b2b205864da075**
Авторизація з введенням неправильного логіну/паролю	Login	Button1_Click	Логін, пароль	Повідомлення про неправильне введення
Автоматичне завантаження записів таблиці бази даних	Product	FillGridView() FillUpdateGridView() FilldeleteGridView() FillPaidGridView() FillUnpaidGridView() FillSellGridView()	Відсутні	Виведення записів таблиці в інтерфейс

Продовження таблиці 4.1.

Вибір запису талиці Автоматичне заповнення текстових контейнерів атрибутами запису таблиці	Product	**itemlist_CellContentClick()** **u_dataGridView _CellContentClick()** **d_item_dataGridView _CellContentClick()** **paid_dataGridView1 _CellContentClick()** **unp_dataGridView _CellContentClick()** **sell_list_CellContentClick()**	Вибір запису	Виведення атрибутів запису
Зміна структури інтерфейсу при виборі режиму роботи з таблицями	Product	**add_Click** **update_Click** **delete_Click** **paid_orders_Click** **unpaid_orders_Click**	Відсутні	Вікно програми виводить нові елементи інтерфейсу
Додавання запису таблиці товарів	Product	additem_Click	**Заповнені текстові контейнери атрибутів запису**	**Повідомлення про успішну дію.** **Виклик автоматичного оновлення таблиці**
Оновлення запису таблиці товарів	Product	u_itemBtn_Click	Вибір запису таблиці. Заповнені текстові контейнери атрибутів запису	Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці.
Видалення запису таблиці товарів	Product	del_item_btn_Click	Вибір запису таблиці.	**Повідомлення про успішну дію.** **Виклик автоматичного оновлення таблиці.**
Зміна статусу запису в таблицях поставлених та непоставлених замовлень	Product	**unp_make_btn_Click** **make_unPaid_btn_Click**	Вибір запису	**Повідомлення про успішну дію.** **Виклик автоматичного оновлення таблиці**

Кінець таблиці 4.1.

1	2	3	4	5
Видалення запису в таблицях поставлених та непоставлених замовлень	Product	**unp_cancelorder_btn_Click cancel_order_btn_Click**	Вибір запису	**Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці.**
Створення файлу PDF з таблицею	Product	exportgridtopdf	Контейнер з масивом даних Назва файлу	Створений файл з встановленою назвою.

Інтеграційне тестування даного додатку характеризується відсутністю обміну даних між модулями. Функції взаємодії представлені тільки викликами нового вікна. Тому одночасно відбувалося і інтеграційне та системне тестування, частина, яких наведена в таблиці 4.2

Таблиця 4.2 – Тестові сценарії системних тесів

Іденти-фікатор	Назва сценарію	Вихідні дані	Очікуваний результат
1	2	3	4
ST-1	Створення запису постачальника	1. Користувач додаток. 2. Користувач вводить логін та пароль. 3. Користувач натиснув кнопку “Товар”. 4. Користувач заповнює контейнери, підписані рядками «Назва», «Представник», «Телефон», «Постачання» та натискає кнопку «Додати»	1. Користувачу надсилається вікно авторизації. 2. Користувачу надсилається основне робоче вікно в режимі додавання запису до таблиці. У інтерфейс автоматично виводиться таблиця з записами про товари.

Продовження таблиці 4.2.

1	2	3	4
			3. Система змінює елементи заповнення даних, їх підписи та виводить нову таблицю з даними про постачальників. 4. Система виводить повідомлення про успішну операцію та виводить оновлену таблицю.
ST-2	Створення PDF-файлу з відсортованою таблицею постачальників	1. Користувач через інтерфейс виконує сортування колонок таблиці натискаючи на їх заголовки. 2. Користувач натиснув кнопку “Створити PDF” 3. Користувач вказує місце збереження. 4. Користувач натиснув кнопку “Зберегти”. 5. Користувач натиснув кнопку “Підтвердити” 6. Користувач натиснув наступну кнопку “Підтвердити”	1. Система відображає змінену таблицю. 2. Система викликає діалогове вікно Windows, де автомаично введена назва файлу «Звіт по Постачальникам». 3. Файлова система відображає місце збереження. 4. Система зберігає новий файл та закриває діалогове вікно.

4.3 Аналіз результатів тестування системи

Результати модульного тестування програмного забезпечення наведено у таблиці 4.3.

Таблиця 4.3 – Результати модульного тестування

Назва сценарію	Метод	Вхідні дані	Отриманий результат	Результат
1	2	3	4	5
Авторизація користувача	Button1_Click	Логін, пароль	Поява основного вінка програми	**Правильно**
Хешування введених даних	MD5Hash	Текс User	8f9bfe9d1345237 cb3b2b205864da075	**Правильно**
Авторизація з введенням неправильного логіну/паролю	Button1_Click	Логін, пароль	Повідомлення про неправильне введення	**Правильно**
Автоматичне завантаження записів таблиці бази даних	**FillGridView()** **FillUpdateGridView()** **FilldeleteGridView()** **FillPaidGridView()** **FillUnpaidGridView()** **FillSellGridView()**	Відсутні	Виведення записів таблиці в інтерфейс	**Правильно**
Зміна статусу запису в таблицях поставлених та непоставлених замовлень	**unp_make_btn_Click** **make_unPaid_btn _Click**	Вибір запису	**Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці**	**Правильно**

Продовження таблиці 4.3

1	2	3	4	5
Зміна статусу запису в таблицях поставлених та непоставлених замовлень	**unp_make_btn _Click** **make_unPaid_btn _Click**	Вибір запису	**Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці**	**Правильно**
Вибір запису талиці Автоматичне заповнення текстових контейнерів атрибутами запису таблиці	**itemlist_CellContentClick()** **u_dataGridView _CellContentClick()** **d_item_dataGridView _CellContentClick()** **paid_dataGridView1 _CellContentClick()** **unp_dataGridView _CellContentClick()** **sell_list_CellContentClick()**	Вибір запису	Виведення атрибутів запису	**Правильно**
Зміна структури інтерфейсу при виборі режиму роботи з таблицями	**add_Click** **update_Click** **delete_Click** **paid_orders_Click** **unpaid_orders_Click**	Відсутні	Вікно програми виводить нові елементи інтерфейсу	**Правильно**
Додавання запису таблиці товарів	additem_Click	Заповнені текстові контейнери атрибутів запису	Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці	**Правильно**
Оновлення запису таблиці товарів	u_itemBtn_Click	Вибір запису таблиці. Заповнені текстові контейнери	Повідомлення про успішну дію. Виклик автоматичного оновлення	**Правильно**

Кінець таблиці 4.3.

1	2	3	4	5
		атрибутів запису	таблиці.	
Видалення запису таблиці товарів	del_item_btn_ Click	Вибір запису таблиці.	**Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці..**	**Правильно**
Зміна статусу запису в таблицях поставлених та непоставлених замовлень	**unp_make_btn _Click** **make_unPaid_ btn_Click**	Вибір запису	**Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці**	**Правильно**
Видалення запису в таблицях поставлених та непоставлених замовлень	**unp_cancelorder _btn_Click** **cancel_order_bt n_Click**	Вибір запису	Вибір запису	**Правильно**
Створення файлу PDF з таблицею	**exportgridtopdf**	Контейнер з масивом даних Назва файлу	Створений файл з встановленою назвою.	**Правильно**
ST-1	Створення запису постачальника	Текст заповнених форм атрибутів	**Повідомлення про успішну дію. Виклик автоматичного оновлення таблиці**	**Правильно**
ST-2	Створення PDF-файлу з відсортованою таблицею постачальників	Назва файлу Місце його збереження	Збережений файл з назвою	**Правильно**

В даному розділі було обрано та аргументовано підходи до тестування системи. Прийняте рішення протестувати систему на різних рівнях за

допомогою модульного, інтеграційного та системного тестування. Для виконання тестування були обрані інструменти для автоматизації. Результати отримані після виконання серії тестувань на всіх рівнях програми свідчить, що розроблений програмний додаток відповідає визначеним функціональним вимогами, є працездатною та може бути використана звичайними користувачами.

ВИСНОВКИ

У процесі виконання дипломного проекту було проведено аналіз та огляд діяльності складу та складського обліку. Були розглянуті приклади існуючого програмного забезпечення в області складського обліку, їх можливості, особливості, переваги та недоліки. На основі отриманої інформації відбулося формування вимоги до розроблюваного програмного забезпечення та створено технічне завдання.

Після визначення специфіки розроблюваного програмного забезпечення було проведено дослідження існуючих архітектур систем, які працюють з базами даних. На основі дослідження формувалася архітектура розроблюваного проектного додатку з урахування його специфіки. В процесі виділено основні елементи системи та визначена специфіка взаємодії між ними. Були розглянуті основні види баз даних за різновидами моделей даних. В результаті обрана реляційна база даних на основі використання Microsoft SQL Server. Проводилося проектування інтерфейсу, визначені його функціональні елементи та логіка використання форм. Для розробки проектного програмного забезпечення приймалося рішення про вибір мови програмування, платформи розробки, засобів створення графічного інтерфейсу користувача та середовища розробки.

Після визначення архітектури програмного було проведене проектування модулів розроблюваного програмного забезпечення, визначена їх структура та взаємодії між ними. Відбулася реалізація логіки додатку з використанням мови програмування, фрагменти коду якої описали виконання основних функцій. На основі цього розроблено керівництво користувача, яке описує роботу з програмним додатком. Були визначені вимоги до технічних та програмних засобів для запуску системи, а також описаний процес розгортання та встановлення системи.

Кінцевим етапом виконання проект став вибір та аргументування підходів до тестування системи. Було вирішено протестувати систему на різних рівнях за

допомогою модульного, інтеграційного та системного тестування. Для виконання тестування обиралися інструменти для автоматизації. Результати отримані після виконання серії тестувань на всіх рівнях програми свідчить, що розроблений програмний додаток відповідає визначеним функціональним вимогами, є працездатною та може бути використана звичайними користувачами.

В результаті виконання дипломного проектування була створена система керування локальної бази даних для обліку матеріальних цінностей на складі, функціонал якої відповідає встановленим вимогам та справно керує при використанні цієї програми звичайним користувачем.

Широкі можливості застосованих у розробці мови програмування C# та платформи розробки .NET Framework, дозволяють подальшу розробку цього програмного додатку з метою його покращення. Серед можливих напрямів продовження роботи над проектом варто відмітити наступні варіанти:

– розширення функціоналу для використання інших баз даних, не закріплених до коду програми;

– розширення функціоналу в області утворення мережевих систем;

– покращення засобів утворення записів про базу даних у текстових файлах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Електронний документообіг: можливості та переваги [Електронний ресурс] // Інтелектуальний сервіс. – 2019. – Режим доступу до ресурсу: https://intelserv.net.ua/news/material/id/529.
2. Тюріна Н. М. Логістика: Навч. посіб. / Н. М.Тюріна, І. В. Гой, І. В. Бабій. – К.: «Центр учбової літератури», 2015. – 392 с.
3. Дудар Т. Г., Волошин Р. В. Основи логістики. Навч. посіб. -К.: Центр учбової літератури, 2012. - 176 с.
4. Посадова інструкція начальника складу [Електронний ресурс] – Режим доступу до ресурсу: https://pomichnyk.org/dlya-yurydychnyh-osib/kadry-ta-dilovodstvo/posadovi-instruktsiji/posadova-instruktsiya-nachalnyka-skladu.html
5. КАДРОВА ЛОГІКА ДЛЯ СЛУЖБИ ЛОГІСТИКИ [Електронний ресурс] // Професійні видання. – 2014. – Режим доступу до ресурсу: https://kadrhelp.com.ua/kadrova-logika-dlya-sluzhbi-logistiki
6. Садовська І. Б. Бухгалтерський облік : навч. посіб. / І. Б. Садовська. - К. : ЦУЛ, 2013. - 688 с
7. ОСОБЛИВОСТІ ТА ПЕРЕВАГИ ПРОГРАМИ BAS ДЛЯ БУХГАЛТЕРСЬКОГО І ТОРГОВОГО ОБЛІКУ [Електронний ресурс] // НЕТСОФТ – Режим доступу до ресурсу: https://www.netsoft.com.ua/articles-soft/bas-news/osobjennosti-i-prjeimushhjestva-programmy-basU.html
8. Чим відрізняється BAS Бухгалтерія від 1С:Підприємство Бухгалтерія? [Електронний ресурс] // Група компаній А4 – Режим доступу до ресурсу: https://a4.com.ua/chim-vidriznyaetsya-bas-buhgalteriya-vid-1spidpriemstvo-buhgalteriya/
9. BAS БУХГАЛТЕРІЯ. ПРОФ [Електронний ресурс] // CONTO – Режим доступу до ресурсу: https://conto.com.ua/ua/products/1c8/bas-bukhgalteriya/
10. BAS БУХГАЛТЕРІЯ. КОРП [Електронний ресурс] // CONTO – Режим доступу до ресурсу: https://conto.com.ua/ua/products/1c8/bas-bukhgalteriya/

11. Усі можливості Торгсофт [Електронний ресурс] // Торгсофт – Режим доступу до ресурсу: https://torgsoft.ua/features/

12. Переваги Торгсофт [Електронний ресурс] // Торгсофт – Режим доступу до ресурсу: https://torgsoft.ua/about/difference/

13. Склад [Електронний ресурс] // Торгсофт – Режим доступу до ресурсу: https://torgsoft.ua/features/sklad/

14. Архітектура бази даних: поняття, визначення, рівні [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://ukr.kagutech.com/4319239-database-architecture-concept-definition-levels

15. Нужний Є. М. Інструментальні засоби електронного офісу : навч. посіб. / Є. М. Нужний, І. В. Клименко, О. О. Акімов. – К. : Центр учбової літератури, 2016. – 296 с.

16. A tour of the C# language [Електронний ресурс] // Microsoft. – 2022. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

17. Що таке .NET і чим займаються .NET-розробники? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://training.epam.ua/#!/News/301?lang=ua.

18. Overview of .NET Framework [Електронний ресурс] // Microsoft. – 2021. – Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/dotnet/framework/get-started/overview.

ДОДАТОК А

(Обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту системи автоматизації обліку матеріальних цінностей на складі.

Умовне позначення продукту: складська програма «Складовик».

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення.

Найменування розробки: система автоматизації обліку матеріальних цінностей на складі.

2 Призначення розробки

Програма є складовою бакалаврської дипломної роботи.

Програма призначена для обліку продажів, покупок та наявності товару на складах, обліку постачальників товару, виведення інформації у вигляді електронного документу для позапрограмного подальшого використання у такому форматі.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Програма повинна працювати з однією базою товарів магазину комп’ютерної техніки, переключатися між її частинами (таблицями):

– Товар – містить інформацію про наявний на складі товар, його кількість, поштучну ціну, загальну категорію (комп’ютери, периферійні пристрої, додаткове обладнання та складові частини), тип товару та його модель.

– Постачальники – містить інформацію про компанії, які постачають продукцію. Вказується назва компанії, її представник, номер телефону, за яким звертатися, та категорії продукції, яка постачається.

– Замовлення – описуються замовлення на постачання продукції. Вказується дата формування замовлення, дата прибуття товару на склад, постачальник, назва моделі товару, його тип, поштучна ціна, кількість та стан виконання постачання (тобто чи було прибуття вантажу на склад).

– Збут – відображає об’єми товару, який покинув склад. Вказується дата, назва моделі товару, його тип, та кількість товару, вилученого зі складу за день вказаної дати.

Додаток має надавати доступ до перегляду змісту та операцій додавання, оновлення та видалення записів з поточної таблиці бази даних. Для таблиці постачальників окремо передбачити функції зміни статусу виконання замовлень: як виконаних, так і невиконаних. Також інтерфейс має надавати можливість виконувати сортування записів таблиці.

Програмний додаток має дозволяти виведення поточної таблиці у файл формату PDF з можливістю його зберігання на комп’ютері.

3.2 Вимоги до надійності

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких наведено нижче:

– організацією безперебійного живлення технічних засобів;

– достатнім рівнем кваліфікації користувача в предметній області використання програми та в користуванні комп’ютерною технікою.

Час відновлення після відмови, викликаного збоєм електроживлення технічних засобів (іншими зовнішніми факторами), які не фатальним збоєм (не крах) операційної системи, не повинно перевищувати часу, необхідного на перезавантаження операційної системи і запуск програми, за умови дотримання умов експлуатації технічних і програмних засобів.

Час відновлення після відмови, викликаного несправністю технічних засобів, фатальним збоєм (крахом) операційної системи, не повинно перевищувати часу, необхідного на усунення несправностей технічних засобів і перевстановлення програмних засобів.

Відмови програми можливі внаслідок некоректних дій оператора (користувача) при взаємодії з операційною системою. Щоб уникнути виникнення відмов програми за вказаною вище причини слід забезпечити роботу кінцевого користувача без надання йому адміністративних привілеїв.

3.3 Вимоги до складу та параметрів технічних засобів

З урахуванням додаткового програмного забезпеченням, мінімальні системні вимоги наступні:

- 64-розрядний процесор з частотою 1 ГГц;
- 1 Гб оперативної пам’яті;
- 10 Гб дискового простору.

Периферійні пристрої: монітор, клавіатура, комп’ютерна миша.

3.4 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати на Windows 7 SP1 або новішій операційній системі.

Для використання програмного забезпечення необхідно:

- операційна система Windows;
- .NET Framework версії 4.6.2 або новіше;
- Microsoft SQL Server Express 2016 або новіше.

3.5 Вимоги до транспортування та зберігання

Надання продукту та документації відбувається у цифровому вигляді. Умови експлуатації програмного забезпечення збігаються з умовами експлуатації комп’ютерної техніки, на якій буде розміщене ПЗ.

3.6 Спеціальні вимоги

Інтерфейс повинен бути інтуїтивно зрозумілим.

Тексти програм програми повинні бути реалізовані на мові C#. В якості інтегрованої середовища розробки програми повинна бути використане середовище

Microsoft Visual Studio. Системні програмні засоби, що використовуються програмою, повинні бути представлені локалізованої версією операційної системи Windows.

4. Вимоги до програмної документації

В ході розробки програми повинні бути підготовлені:

- текст програми,
- опис програми,
- програма
- технічне завдання
- керівництво користувача

5. Стадії та етапи розробки

Стадії та етапи розробки програмної система для автоматизації публікації, поширення та пошуку кулінарних рецептів подані у таблиці А.1

Таблиця А.1 – Стадії та етапи розробки

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 01.01.22 – 31.01.22	Обґрунтування необхідності розробки програми	Характеристика та опис ПЗ; підстави для розробки і призначення ПЗ; функціональні вимоги до розроблюваної системи; порядок контролю і приймання ПЗ.
Ескізний проект 01.02.22 – 15.02.22	Розробка ескізного проекту	Визначення структури вхідних і вихідних даних; попередній вибір технологій; визначення потрібних алгоритмів.
Технічний проект 16.02.22 – 28.02.22	Розробка технічного	Затвердження структури вхідних і вихідних даних; розробка алгоритмів;

	проекту	розробка структури програми; вибір технологій.
Робочий проект 01.03.22 – 10.04.22	Розробка програмного забезпечення	Реалізація програмного забезпечення; виправлення помилок; тестування.
Розробка програмної документації 11.04.21 – 20.04.21	Розробка документації для програмного забезпечення	Розробка документації користувача, інструкції по запуску та зупинці ПЗ
Тестування системи 21.04.21 – 30.04.21	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Здача проекту	Здача програми замовнику	Передача вихідного коду та документації замовнику

6. Порядок контролю та приймання

Контроль здійснюється кінцевим користувачем-замовником, якого буде задіяно по закінченні етапу тестування.

Після закінчення розробки програмного продукту має бути проведено тестування на виконання передбачених функцій та на захист від введення некоректних параметрів.

ДОДАТОК Б

(Обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Клас Start.cs:

```
    static class Start
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Login());
        }
    }
```

Клас/форма Login.cs:

```
public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                string user = MD5Hash(Textbox1.Text.Trim());
                string user1 = "8f9bfe9d1345237cb3b2b205864da075";
                string password = MD5Hash(Textbox2.Text.Trim());
                string password1 = "5f4dcc3b5aa765d61d8327deb882cf99";
                if (password == password1 && user == user1)
                {
                    Product product = new Product();
                    product.Show();
                    this.Hide();
                }

                else
                {
                    MessageBox.Show("Неправильний пароль та/або користувач");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }

        public static string MD5Hash(string input)
        {
            StringBuilder hash = new StringBuilder();
            MD5CryptoServiceProvider md5provider = new
MD5CryptoServiceProvider();
            byte[] bytes = md5provider.ComputeHash(new
UTF8Encoding().GetBytes(input));

            for (int i = 0; i < bytes.Length; i++)
```

```
            {
                hash.Append(bytes[i].ToString("x2"));
            }
            return hash.ToString();
        }

        private void close_btn_Click(object sender, EventArgs e)
        {
            if (System.Windows.Forms.Application.MessageLoop)
            {
                System.Windows.Forms.Application.Exit();
            }
            else
            {
                System.Environment.Exit(1);
            }
        }
    }
}
```

Клас/форма Product.cs:

```
public partial class Product : Form
    {
        public const int WM_NCLBUTTONDOWN = 0xA1;
        public const int HT_CAPTION = 0x2;

        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern int SendMessage(IntPtr hWnd, int Msg, int
wParam, int lParam);
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern bool ReleaseCapture();


        public Product()
        {
            InitializeComponent();
            slide_panel.Height = add.Height;
            slide_panel.Top = add.Top;
            additem_panel.BringToFront();
        }


        private void Product_Load(object sender, EventArgs e)
        {
            itemcode.Enabled = false;
            itemcode.Text = "Id Auto Number";
            FillGridView();

            u_itemcodeTxt.Enabled = false;
            u_itemcodeTxt.Text = "Id Auto Number";
            FillUpdateGridView();

            d_itemcodeTxt.Enabled = false;
            d_itemcodeTxt.Text = "Id Auto Number";
            FilldeleteGridView();
```

```
    p_order_idTxt.Enabled = false;
    p_order_idTxt.Text = "Id Auto Number";
    FillPaidGridView();

    unp_orderidTxt.Enabled = false;
    unp_orderidTxt.Text = "Id Auto Number";
    FillUnpaidGridView();

    s_id.Enabled = false;
    s_id.Text = "Id Auto Number";
    FillSellGridView();
}

private void close_btn_Click(object sender, EventArgs e)
{
   if (System.Windows.Forms.Application.MessageLoop)
   {
        System.Windows.Forms.Application.Exit();
   }
   else
   {
        System.Environment.Exit(1);
   }
}

private void add_Click(object sender, EventArgs e)
{
    slide_panel.Height = add.Height;
    slide_panel.Top = add.Top;
    additem_panel.BringToFront();
    FillGridView();
    model.Clear();
    part.Clear();
    price.Clear();
    instock.Clear();
    comboBox1.SelectedIndex = -1;
    itemcode.Enabled = false;
    itemcode.Text = "Id Auto Number";
}

private void update_Click(object sender, EventArgs e)
{
    slide_panel.Height = update.Height;
    slide_panel.Top = update.Top;
    updateitems_panel.BringToFront();
    FillUpdateGridView();
    u_modelTxt.Clear();
    u_partTxt.Clear();
    u_priceTxt.Clear();
    u_stockTxt.Clear();
    u_typeCombo.SelectedIndex = -1;
    u_itemcodeTxt.Enabled = false;
    u_itemcodeTxt.Text = "Id Auto Number";
}

private void delete_Click(object sender, EventArgs e)
{
    slide_panel.Height = delete.Height;
```

```
            slide_panel.Top = delete.Top;
            deleteitem_panel.BringToFront();
            FilldeleteGridView();
            d_modelTxt.Clear();
            d_partTxt.Clear();
            d_priceTxt.Clear();
            d_instockTxt.Clear();
            d_typeCombo.SelectedIndex = -1;
            d_itemcodeTxt.Enabled = false;
            d_itemcodeTxt.Text = "Id Auto Number";
        }

        private void paid_orders_Click(object sender, EventArgs e)
        {
            slide_panel.Height = paid_orders.Height;
            slide_panel.Top = paid_orders.Top;
            paid_orders_panel.BringToFront();
            FillPaidGridView();
            p_order_detailsTxt.Clear();
            p_partTxt.Clear();
            p_order_priceTxt.Clear();
            p_order_paidTxt.Clear();
            p_order_idTxt.Enabled = false;
            p_order_idTxt.Text = "Id Auto Number";
        }

        private void unpaid_orders_Click(object sender, EventArgs e)
        {

            slide_panel.Height = unpaid_orders.Height;
            slide_panel.Top = unpaid_orders.Top;
            unp_order_panel.BringToFront();
            FillUnpaidGridView();
            unp_orderidTxt.Enabled = false;
            unp_orderidTxt.Text = "Id Auto Number";
            unp_orderdetailsTxt.Clear();
            unp_partTxt.Clear();
            unp_priceTxt.Clear();
            unp_ispaidTxt.Clear();
        }

        private void sell_btn_Click(object sender, EventArgs e)
        {
            slide_panel.Height = sell_btn.Height;
            slide_panel.Top = sell_btn.Top;
            sell_panell.BringToFront();
            FillSellGridView();
            s_id.Clear();
            s_date.Clear();
            s_name.Clear();
            s_type.Clear();
            s_ammount.Clear();
        }

        private void additem_Click(object sender, EventArgs e)
        {
            if(model.Text!="" && part.Text!="" && comboBox1.Text!="" &&
price.Text!="" && instock.Text != "")
```

```
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "INSERT INTO Товар(Model, Type, Category,
Price, Amount) VALUES(@Model, @Type, @Category, @Price, @Amount)";
                    double Price = double.Parse(price.Text);
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Model",
model.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
part.Text.ToString());
                    cmd.Parameters.AddWithValue("@Category",
comboBox1.Text.ToString());
                    cmd.Parameters.AddWithValue("@Price", Price);
                    cmd.Parameters.AddWithValue("@Amount",
instock.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис додано!");
                    model.Clear();
                    part.Clear();
                    price.Clear();
                    instock.Clear();
                    comboBox1.SelectedIndex = -1;
                    FillGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповність всі поля");
            }
        }

        void FillGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            itemlist.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Товар", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                itemlist.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString());
            }
            dr.Close();
            conn.Close();
```

```
        }

        private void itemlist_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if(e.RowIndex >= 0)
            {
                DataGridViewRow row = this.itemlist.Rows[e.RowIndex];
                itemcode.Text = row.Cells[0].Value.ToString();
                model.Text = row.Cells[1].Value.ToString();
                part.Text = row.Cells[2].Value.ToString();
                comboBox1.Text = row.Cells[3].Value.ToString();
                price.Text = row.Cells[4].Value.ToString();
                instock.Text = row.Cells[5].Value.ToString();
            }
        }

        private void u_itemBtn_Click(object sender, EventArgs e)
        {
            if (u_modelTxt.Text != "" && u_partTxt.Text != "" &&
u_typeCombo.Text != "" && u_priceTxt.Text != "" && u_stockTxt.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Товар set Model = @Model,Type =
@Type,Category = @Category,Price =@Price, Amount = @Amount where ID = '"+
u_itemcodeTxt.Text +"' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    double Price = double.Parse(u_priceTxt.Text);
                    cmd.Parameters.AddWithValue("@Model",
u_modelTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
u_partTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Category",
u_typeCombo.Text.ToString());
                    cmd.Parameters.AddWithValue("@Price", Price);
                    cmd.Parameters.AddWithValue("@Amount",
u_stockTxt.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис оновлено");
                    u_itemcodeTxt.Clear();
                    u_itemcodeTxt.Enabled = false;
                    u_itemcodeTxt.Text = "Id Auto Number";
                    u_modelTxt.Clear();
                    u_partTxt.Clear();
                    u_priceTxt.Clear();
                    u_stockTxt.Clear();
                    u_typeCombo.SelectedIndex = -1;
                    FillUpdateGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
```

```
                }
            }
            else
            {
                MessageBox.Show("Виберіть запис");
            }

        }

        void FillUpdateGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            u_dataGridView.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Товар", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                u_dataGridView.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString());
            }
        }

        private void u_dataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = this.u_dataGridView.Rows[e.RowIndex];
                u_itemcodeTxt.Text = row.Cells[0].Value.ToString();
                u_modelTxt.Text = row.Cells[1].Value.ToString();
                u_partTxt.Text = row.Cells[2].Value.ToString();
                u_typeCombo.Text = row.Cells[3].Value.ToString();
                u_priceTxt.Text = row.Cells[4].Value.ToString();
                u_stockTxt.Text = row.Cells[5].Value.ToString();
            }
        }

        private void del_item_btn_Click(object sender, EventArgs e)
        {
            if (d_modelTxt.Text != "" && d_partTxt.Text != "" &&
d_typeCombo.Text != "" && d_priceTxt.Text != "" && d_instockTxt.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Товар where ID = '" +
d_itemcodeTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
```

```
                    MessageBox.Show("The item has been successfully
deleted!");
                    d_itemcodeTxt.Clear();
                    d_itemcodeTxt.Enabled = false;
                    d_itemcodeTxt.Text = "Id Auto Number";
                    d_modelTxt.Clear();
                    d_partTxt.Clear();
                    d_priceTxt.Clear();
                    d_instockTxt.Clear();
                    d_typeCombo.SelectedIndex = -1;
                    FilldeleteGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        void FilldeleteGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            d_item_dataGridView.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Товар", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                d_item_dataGridView.Rows.Add(dr[0].ToString(),
dr[1].ToString(), dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString());
            }
        }

        private void d_item_dataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row =
this.d_item_dataGridView.Rows[e.RowIndex];
                d_itemcodeTxt.Text = row.Cells[0].Value.ToString();
                d_modelTxt.Text = row.Cells[1].Value.ToString();
                d_partTxt.Text = row.Cells[2].Value.ToString();
                d_typeCombo.Text = row.Cells[3].Value.ToString();
                d_priceTxt.Text = row.Cells[4].Value.ToString();
                d_instockTxt.Text = row.Cells[5].Value.ToString();
            }
        }

        private void make_unPaid_btn_Click(object sender, EventArgs e)
        {
```

```
            if (p_order_idTxt.Text != ""&& p_order_detailsTxt.Text !="")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Замовлення set Done = '" + "-" +
"'where ID = '" + p_order_idTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення позначено як недоставлене");
                    p_order_idTxt.Clear();
                    p_order_idTxt.Enabled = false;
                    p_order_idTxt.Text = "Id Auto Number";
                    p_order_detailsTxt.Clear();
                    p_partTxt.Clear();
                    p_order_priceTxt.Clear();
                    p_order_AmmTxt.Clear();
                    p_order_paidTxt.Clear();
                    FillPaidGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("First...u should select an order to make
changes");
            }
        }

        void FillPaidGridView()
        {
            string status = "+";
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            paid_dataGridView1.Rows.Clear();
            SqlCommand cmd = new SqlCommand("select * from Замовлення where
Done = '" + status + "' ", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                paid_dataGridView1.Rows.Add(dr[0].ToString(),
dr[1].ToString(), dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString(), dr[6].ToString(),
dr[7].ToString(), dr[8].ToString());
            }
            dr.Close();
            conn.Close();
        }
```

```
        private void paid_dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row =
this.paid_dataGridView1.Rows[e.RowIndex];
                p_order_idTxt.Text = row.Cells[0].Value.ToString();
                p_order_detailsTxt.Text = row.Cells[4].Value.ToString();
                p_partTxt.Text = row.Cells[5].Value.ToString();
                p_order_priceTxt.Text = row.Cells[6].Value.ToString();
                p_order_AmmTxt.Text = row.Cells[7].Value.ToString();
                p_order_paidTxt.Text = row.Cells[8].Value.ToString();
            }
        }

        private void cancel_order_btn_Click(object sender, EventArgs e)
        {
            if (p_order_idTxt.Text != ""&& p_order_detailsTxt.Text !="")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Замовлення where ID = '" +
p_order_idTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення відмінено та видалено");
                    p_order_idTxt.Clear();
                    p_order_idTxt.Enabled = false;
                    p_order_idTxt.Text = "Id Auto Number";
                    p_order_detailsTxt.Clear();
                    p_partTxt.Clear();
                    p_order_priceTxt.Clear();
                    p_order_paidTxt.Clear();
                    FillPaidGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        private void unp_make_btn_Click(object sender, EventArgs e)
        {
            if (unp_orderidTxt.Text != "" && unp_orderdetailsTxt.Text !="")
            {
                try
                {
```

```
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Замовлення set Done = '" + "+" +
"'where ID = '" + unp_orderidTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення позначено як доставлене");
                    unp_orderidTxt.Clear();
                    unp_orderidTxt.Enabled = false;
                    unp_orderidTxt.Text = "Id Auto Number";
                    unp_orderdetailsTxt.Clear();
                    unp_partTxt.Clear();
                    unp_priceTxt.Clear();
                    unp_order_AmmTxt.Clear();
                    unp_ispaidTxt.Clear();
                    FillUnpaidGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        void FillUnpaidGridView()
        {
            string status = "-";
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            unp_dataGridView.Rows.Clear();
            SqlCommand cmd = new SqlCommand("select * from Замовлення where
Done = '" + status + "' ", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                unp_dataGridView.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString(), dr[6].ToString(),
dr[7].ToString(), dr[8].ToString());
            }
            dr.Close();
            conn.Close();
        }

        private void unp_dataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
```

```
            DataGridViewRow row = this.unp_dataGridView.Rows[e.RowIndex];
            unp_orderidTxt.Text = row.Cells[0].Value.ToString();
            unp_orderdetailsTxt.Text = row.Cells[4].Value.ToString();
            unp_partTxt.Text = row.Cells[5].Value.ToString();
            unp_priceTxt.Text = row.Cells[6].Value.ToString();
            unp_order_AmmTxt.Text = row.Cells[7].Value.ToString();
            unp_ispaidTxt.Text = row.Cells[8].Value.ToString();
        }
    }

    private void unp_cancelorder_btn_Click(object sender, EventArgs e)
    {
        if (unp_orderidTxt.Text != "" && unp_orderdetailsTxt.Text != "")
        {
            try
            {
                SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                string query = "delete from Замовлення where ID = '" +
p_order_idTxt.Text + "' ";
                SqlCommand cmd = new SqlCommand(query, conn);
                conn.Open();
                cmd.ExecuteNonQuery();
                conn.Close();
                MessageBox.Show("Замовлення відмінено та видалено");
                unp_orderidTxt.Clear();
                unp_orderidTxt.Enabled = false;
                unp_orderidTxt.Text = "Id Auto Number";
                unp_orderdetailsTxt.Clear();
                unp_partTxt.Clear();
                unp_priceTxt.Clear();
                unp_ispaidTxt.Clear();
                FillUnpaidGridView();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
        else
        {
            MessageBox.Show("Виберіть рядок");
        }
    }

    private void sell_list_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        DataGridViewRow row = this.sell_list.Rows[e.RowIndex];
        s_id.Text = row.Cells[0].Value.ToString();
        s_date.Text = row.Cells[1].Value.ToString();
        s_name.Text = row.Cells[2].Value.ToString();
        s_type.Text = row.Cells[3].Value.ToString();
        s_ammount.Text = row.Cells[4].Value.ToString();
    }

    private void s_add_Click(object sender, EventArgs e)
```

```
        {
            if (s_id.Text != "" && s_date.Text != "" && s_name.Text != "" &&
s_type.Text != "" && s_ammount.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "INSERT INTO Збут(Date, Model, Type,
Amount) VALUES(@Date, @Model, @Type, @Amount)";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Date",
s_date.Text.ToString());
                    cmd.Parameters.AddWithValue("@Model",
s_name.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
s_type.Text.ToString());
                    cmd.Parameters.AddWithValue("@Amount",
s_ammount.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис додано");
                    s_id.Clear();
                    s_date.Clear();
                    s_name.Clear();
                    s_type.Clear();
                    s_ammount.Clear();
                    FillSellGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповніть всі поля");
            }
        }

        private void s_delete_Click(object sender, EventArgs e)
        {
            if (s_id.Text != "" && s_date.Text != "" && s_name.Text != "" &&
s_type.Text != "" && s_ammount.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Збут where ID = '" +
s_id.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
```

```
                    MessageBox.Show("Запис видалено");
                    s_id.Clear();
                    s_date.Clear();
                    s_name.Clear();
                    s_type.Clear();
                    s_ammount.Clear();
                    FillSellGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        private void s_update_Click(object sender, EventArgs e)
        {
            if (s_id.Text != "" && s_date.Text != "" && s_name.Text != "" &&
s_type.Text != "" && s_ammount.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Збут set Date = @Date,Model =
@Model,Type = @Type, Amount= @Amount where ID = '" + s_id.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Date",
s_date.Text.ToString());
                    cmd.Parameters.AddWithValue("@Model",
s_name.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
s_type.Text.ToString());
                    cmd.Parameters.AddWithValue("@Amount",
s_ammount.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис оновлено");
                    s_id.Clear();
                    s_date.Clear();
                    s_name.Clear();
                    s_type.Clear();
                    s_ammount.Clear();
                    FillSellGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
```

```
                MessageBox.Show("Заповніть всі поля");
            }
        }

        void FillSellGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            sell_list.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Збут", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                sell_list.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString());
            }
            dr.Close();
            conn.Close();
        }
        public void exportgridtopdf(DataGridView dgw, string filename)
        {
            BaseFont bf = BaseFont.CreateFont(Application.StartupPath +
"\\Arial.ttf", BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
            PdfPTable pdftable = new PdfPTable(dgw.Columns.Count);
            pdftable.DefaultCell.Padding = 3;
            pdftable.WidthPercentage = 100;
            pdftable.HorizontalAlignment = Element.ALIGN_LEFT;
            pdftable.DefaultCell.BorderWidth = 1;

            iTextSharp.text.Font text = new iTextSharp.text.Font(bf,
iTextSharp.text.Font.DEFAULTSIZE, iTextSharp.text.Font.NORMAL);
            foreach (DataGridViewColumn column in dgw.Columns)
            {
                PdfPCell cell = new PdfPCell(new Phrase(column.HeaderText,
text));
                cell.BackgroundColor = new iTextSharp.text.BaseColor(240,
240, 240);
                pdftable.AddCell(cell);
            }

            foreach (DataGridViewRow row in dgw.Rows)
            {
                foreach (DataGridViewCell cell in row.Cells)
                {
                    pdftable.AddCell(new Phrase(cell.Value.ToString(),
text));
                }
            }

            var savefiledialoge = new SaveFileDialog();
            savefiledialoge.FileName = filename;
            savefiledialoge.DefaultExt = ".pdf";
            if (savefiledialoge.ShowDialog() == DialogResult.OK)
            {
```

```
                using (FileStream stream = new
FileStream(savefiledialoge.FileName, FileMode.Create))
                {
                    Document pdfdoc = new Document(PageSize.A4, 10f, 10f,
10f, 0f);
                    PdfWriter.GetInstance(pdfdoc, stream);
                    pdfdoc.Open();
                    pdfdoc.Add(pdftable);
                    pdfdoc.Close();
                    stream.Close();
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            exportgridtopdf(itemlist, "Звітність по Товару");
        }

        private void button2_Click(object sender, EventArgs e)
        {
            exportgridtopdf(unp_dataGridView, "Звіт по неоплаченм
Замовленням");
        }

        private void button3_Click(object sender, EventArgs e)
        {
            exportgridtopdf(paid_dataGridView1, "Звіт по оплаченм
Замовленням");
        }

        private void button7_Click(object sender, EventArgs e)
        {
            exportgridtopdf(sell_list, "Звіт по Збуту");
        }

        private void button9_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Suppliers supp = new Suppliers();
            supp.Show();
            this.Close();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            Orders ord = new Orders();
            ord.Show();
        }

        private void Product_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
```

```
                ReleaseCapture();
                SendMessage(Handle, WM_NCLBUTTONDOWN, HT_CAPTION, 0);
            }
        }
    }
```

Клас/форма Suppliers.cs:

```
public partial class Suppliers : Form
    {
        public const int WM_NCLBUTTONDOWN = 0xA1;
        public const int HT_CAPTION = 0x2;

        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern int SendMessage(IntPtr hWnd, int Msg, int
wParam, int lParam);
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern bool ReleaseCapture();

        public Suppliers()
        {
            InitializeComponent();
            slide_panel.Height = add.Height;
            slide_panel.Top = add.Top;
            additem_panel.BringToFront();
        }

        private void Suppliers_Load(object sender, EventArgs e)
        {
            FillGridView();

            FillUpdateGridView();

            FilldeleteGridView();

            p_order_idTxt.Enabled = false;
            p_order_idTxt.Text = "Id Auto Number";
            FillPaidGridView();

            unp_orderidTxt.Enabled = false;
            unp_orderidTxt.Text = "Id Auto Number";
            FillUnpaidGridView();

            s_id.Enabled = false;
            s_id.Text = "Id Auto Number";
            FillSellGridView();

        }

        private void close_btn_Click(object sender, EventArgs e)
        {
            if (System.Windows.Forms.Application.MessageLoop)
            {
                System.Windows.Forms.Application.Exit();
            }
            else
            {
```

```
            System.Environment.Exit(1);
        }
    }

    private void add_Click(object sender, EventArgs e)
    {
        slide_panel.Height = add.Height;
        slide_panel.Top = add.Top;
        additem_panel.BringToFront();
        FillGridView();
        itemcode.Clear();
        model.Clear();
        part.Clear();
        richTextBox2.Clear();
    }

    private void sell_btn_Click(object sender, EventArgs e)
    {
        slide_panel.Height = sell_btn.Height;
        slide_panel.Top = sell_btn.Top;
        sell_panell.BringToFront();
        FillSellGridView();
        s_id.Clear();
        s_date.Clear();
        s_name.Clear();
        s_type.Clear();
        s_ammount.Clear();
    }

    private void update_Click(object sender, EventArgs e)
    {
        slide_panel.Height = update.Height;
        slide_panel.Top = update.Top;
        updateitems_panel.BringToFront();
        FillUpdateGridView();
        u_modelTxt.Clear();
        u_partTxt.Clear();
        u_itemcodeTxt.Clear();
        richTextBox3.Clear();
        u_label_id.Text = "0";
    }

    private void delete_Click(object sender, EventArgs e)
    {
        slide_panel.Height = delete.Height;
        slide_panel.Top = delete.Top;
        deleteitem_panel.BringToFront();
        FilldeleteGridView();
        d_itemcodeTxt.Clear();
        d_modelTxt.Clear();
        d_partTxt.Clear();
        richTextBox1.Clear();
    }

    private void paid_orders_Click(object sender, EventArgs e)
    {
        slide_panel.Height = paid_orders.Height;
        slide_panel.Top = paid_orders.Top;
```

```
            paid_orders_panel.BringToFront();
            FillPaidGridView();
            p_order_detailsTxt.Clear();
            p_partTxt.Clear();
            p_order_priceTxt.Clear();
            p_order_paidTxt.Clear();
            p_order_idTxt.Enabled = false;
            p_order_idTxt.Text = "Id Auto Number";
        }

        private void unpaid_orders_Click(object sender, EventArgs e)
        {

            slide_panel.Height = unpaid_orders.Height;
            slide_panel.Top = unpaid_orders.Top;
            unp_order_panel.BringToFront();
            FillUnpaidGridView();
            unp_orderidTxt.Enabled = false;
            unp_orderidTxt.Text = "Id Auto Number";
            unp_orderdetailsTxt.Clear();
            unp_partTxt.Clear();
            unp_priceTxt.Clear();
            unp_ispaidTxt.Clear();
        }



        private void additem_Click(object sender, EventArgs e)
        {
            if (itemcode.Text != "" && model.Text != "" && part.Text != "" &&
richTextBox2.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "INSERT INTO Постачальники(Name,
Representative, Supply, Phone) VALUES(@Name, @Representative, @Supply,
@Phone)";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Name",
itemcode.Text.ToString());
                    cmd.Parameters.AddWithValue("@Representative",
model.Text.ToString());
                    cmd.Parameters.AddWithValue("@Phone",
part.Text.ToString());
                    cmd.Parameters.AddWithValue("@Supply",
richTextBox2.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис додано");
                    itemcode.Clear();
                    model.Clear();
                    part.Clear();
                    richTextBox2.Clear();
                    FillGridView();
```

```
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповніть всі поля");
            }
        }

        void FillGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            itemlist.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Постачальники",
conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                itemlist.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString());
            }
            dr.Close();
            conn.Close();
        }

        private void itemlist_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = this.itemlist.Rows[e.RowIndex];
                itemcode.Text = row.Cells[1].Value.ToString();
                model.Text = row.Cells[2].Value.ToString();
                part.Text = row.Cells[4].Value.ToString();
                richTextBox2.Text = row.Cells[3].Value.ToString();
            }
        }

        private void u_itemBtn_Click(object sender, EventArgs e)
        {
            if (u_itemcodeTxt.Text != "" && u_modelTxt.Text != "" &&
u_partTxt.Text != "" && richTextBox2.Text != "" && u_label_id.Text != "0")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Постачальники set Name =
@Name,Representative = @Representative,Phone = @Phone, Supply= @Supply where
ID = '" + u_label_id.Text + "' ";
```

```
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Name",
u_itemcodeTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Representative",
u_modelTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Phone",
u_partTxt.Text.ToString());
                    cmd.Parameters.AddWithValue("@Supply",
richTextBox3.Text);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис оновлено");
                    u_itemcodeTxt.Clear();
                    u_modelTxt.Clear();
                    u_partTxt.Clear();
                    richTextBox3.Clear();
                    FillUpdateGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }

        }

        void FillUpdateGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            u_dataGridView.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Постачальники",
conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                u_dataGridView.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString());
            }
        }

        private void u_dataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = this.u_dataGridView.Rows[e.RowIndex];
                u_label_id.Text = row.Cells[0].Value.ToString();
                u_itemcodeTxt.Text = row.Cells[1].Value.ToString();
                u_modelTxt.Text = row.Cells[2].Value.ToString();
```

```
                u_partTxt.Text = row.Cells[4].Value.ToString();
                richTextBox3.Text = row.Cells[3].Value.ToString();
            }
        }

        private void del_item_btn_Click(object sender, EventArgs e)
        {
            if (d_modelTxt.Text != "" && d_partTxt.Text != "" &&
richTextBox1.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Представники where Name = '"
+ d_itemcodeTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("The item has been successfully
deleted!");
                    d_itemcodeTxt.Clear();
                    d_itemcodeTxt.Enabled = false;
                    d_itemcodeTxt.Text = "Id Auto Number";
                    d_modelTxt.Clear();
                    d_partTxt.Clear();
                    richTextBox1.Clear();
                    FilldeleteGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        void FilldeleteGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            d_item_dataGridView.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Постачальники",
conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                d_item_dataGridView.Rows.Add(dr[0].ToString(),
dr[1].ToString(), dr[2].ToString(), dr[3].ToString(),dr[4].ToString());
            }
        }
```

```
        private void d_item_dataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row =
this.d_item_dataGridView.Rows[e.RowIndex];
                d_itemcodeTxt.Text = row.Cells[0].Value.ToString();
                d_modelTxt.Text = row.Cells[1].Value.ToString();
                d_partTxt.Text = row.Cells[2].Value.ToString();
            }
        }

        void FillPaidGridView()
        {
            string status = "+";
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            paid_dataGridView1.Rows.Clear();
            SqlCommand cmd = new SqlCommand("select * from Замовлення where
Done = '" + status + "' ", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                paid_dataGridView1.Rows.Add(dr[0].ToString(),
dr[1].ToString(), dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString(), dr[6].ToString(),
dr[7].ToString(), dr[8].ToString());
            }
            dr.Close();
            conn.Close();
        }

        private void paid_dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row =
this.paid_dataGridView1.Rows[e.RowIndex];
                p_order_idTxt.Text = row.Cells[0].Value.ToString();
                p_order_detailsTxt.Text = row.Cells[4].Value.ToString();
                p_partTxt.Text = row.Cells[5].Value.ToString();
                p_order_priceTxt.Text = row.Cells[6].Value.ToString();
                p_order_AmmTxt.Text = row.Cells[7].Value.ToString();
                p_order_paidTxt.Text = row.Cells[8].Value.ToString();
            }
        }

        private void cancel_order_btn_Click(object sender, EventArgs e)
        {
            if (p_order_idTxt.Text != "" && p_order_detailsTxt.Text != "")
            {
                try
                {
```

```
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Замовлення where ID = '" +
p_order_idTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення відмінено та видалено");
                    p_order_idTxt.Clear();
                    p_order_idTxt.Enabled = false;
                    p_order_idTxt.Text = "Id Auto Number";
                    p_order_detailsTxt.Clear();
                    p_partTxt.Clear();
                    p_order_priceTxt.Clear();
                    p_order_paidTxt.Clear();
                    FillPaidGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        private void unp_make_btn_Click(object sender, EventArgs e)
        {
            if (unp_orderidTxt.Text != "" && unp_orderdetailsTxt.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Замовлення set Done = '" + "+" +
"'where ID = '" + unp_orderidTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення позначене як виконанене та
перенесено у відповідну таблицю");
                    unp_orderidTxt.Clear();
                    unp_orderidTxt.Enabled = false;
                    unp_orderidTxt.Text = "Id Auto Number";
                    unp_orderdetailsTxt.Clear();
                    unp_partTxt.Clear();
                    unp_priceTxt.Clear();
                    unp_order_AmmTxt.Clear();
                    unp_ispaidTxt.Clear();
                    FillUnpaidGridView();
                }
                catch (Exception ex)
```

```
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        void FillUnpaidGridView()
        {
            string status = "-";
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            unp_dataGridView.Rows.Clear();
            SqlCommand cmd = new SqlCommand("select * from Замовлення where
Done = '" + status + "' ", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                unp_dataGridView.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString(), dr[6].ToString(),
dr[7].ToString(), dr[8].ToString());
            }
            dr.Close();
            conn.Close();
        }

        private void unp_dataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = this.unp_dataGridView.Rows[e.RowIndex];
                unp_orderidTxt.Text = row.Cells[0].Value.ToString();
                unp_orderdetailsTxt.Text = row.Cells[4].Value.ToString();
                unp_partTxt.Text = row.Cells[5].Value.ToString();
                unp_priceTxt.Text = row.Cells[6].Value.ToString();
                unp_order_AmmTxt.Text = row.Cells[7].Value.ToString();
                unp_ispaidTxt.Text = row.Cells[8].Value.ToString();
            }
        }

        private void make_unPaid_btn_Click(object sender, EventArgs e)
        {
            if (p_order_idTxt.Text != "" && p_order_detailsTxt.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Замовлення set Done = '" + "-" +
"'where ID= '" + p_order_idTxt.Text + "' ";
```

```
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення позначено як невиконане та
відправлено у відповідну таблицю");
                    p_order_idTxt.Clear();
                    p_order_idTxt.Enabled = false;
                    p_order_idTxt.Text = "Id Auto Number";
                    p_order_detailsTxt.Clear();
                    p_partTxt.Clear();
                    p_order_priceTxt.Clear();
                    p_order_AmmTxt.Clear();
                    p_order_paidTxt.Clear();
                    FillPaidGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть запис");
            }
        }

        private void unp_cancelorder_btn_Click(object sender, EventArgs e)
        {
            if (unp_orderidTxt.Text != "" && unp_orderdetailsTxt.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Замовлення where ID = '" +
unp_orderidTxt.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Замовлення відмінено та видалено");
                    unp_orderidTxt.Clear();
                    unp_orderidTxt.Enabled = false;
                    unp_orderidTxt.Text = "Id Auto Number";
                    unp_orderdetailsTxt.Clear();
                    unp_partTxt.Clear();
                    unp_priceTxt.Clear();
                    unp_ispaidTxt.Clear();
                    FillUnpaidGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
```

```
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        public void exportgridtopdf(DataGridView dgw, string filename)
        {
            BaseFont bf = BaseFont.CreateFont(Application.StartupPath +
"\\Arial.ttf", BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
            PdfPTable pdftable = new PdfPTable(dgw.Columns.Count);
            pdftable.DefaultCell.Padding = 3;
            pdftable.WidthPercentage = 100;
            pdftable.HorizontalAlignment = Element.ALIGN_LEFT;
            pdftable.DefaultCell.BorderWidth = 1;

            iTextSharp.text.Font text = new iTextSharp.text.Font(bf,
iTextSharp.text.Font.DEFAULTSIZE, iTextSharp.text.Font.NORMAL);
            foreach (DataGridViewColumn column in dgw.Columns)
            {
                PdfPCell cell = new PdfPCell(new Phrase(column.HeaderText,
text));
                cell.BackgroundColor = new iTextSharp.text.BaseColor(240,
240, 240);
                pdftable.AddCell(cell);
            }

            foreach (DataGridViewRow row in dgw.Rows)
            {
                foreach (DataGridViewCell cell in row.Cells)
                {
                    pdftable.AddCell(new Phrase(cell.Value.ToString(),
text));
                }
            }

            var savefiledialoge = new SaveFileDialog();
            savefiledialoge.FileName = filename;
            savefiledialoge.DefaultExt = ".pdf";
            if (savefiledialoge.ShowDialog() == DialogResult.OK)
            {
                using (FileStream stream = new
FileStream(savefiledialoge.FileName, FileMode.Create))
                {
                    Document pdfdoc = new Document(PageSize.A4, 10f, 10f,
10f, 0f);
                    PdfWriter.GetInstance(pdfdoc, stream);
                    pdfdoc.Open();
                    pdfdoc.Add(pdftable);
                    pdfdoc.Close();
                    stream.Close();
                }
            }
        }

        private void s_add_Click(object sender, EventArgs e)
        {
            if (s_id.Text != "" && s_date.Text != "" && s_name.Text != "" &&
s_type.Text != "" && s_ammount.Text != "")
```

```
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "INSERT INTO Збут(Date, Model, Type,
Amount) VALUES(@Date, @Model, @Type, @Amount)";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Date",
s_date.Text.ToString());
                    cmd.Parameters.AddWithValue("@Model",
s_name.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
s_type.Text.ToString());
                    cmd.Parameters.AddWithValue("@Amount",
s_ammount.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис додано");
                    s_id.Clear();
                    s_date.Clear();
                    s_name.Clear();
                    s_type.Clear();
                    s_ammount.Clear();
                    FillSellGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповніть всі поля");
            }
        }

        private void s_delete_Click(object sender, EventArgs e)
        {
            if (s_id.Text != "" && s_date.Text != "" && s_name.Text != "" &&
s_type.Text != "" && s_ammount.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "delete from Збут where ID = '" +
s_id.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис видалено");
                    s_id.Clear();
                    s_date.Clear();
```

```
                    s_name.Clear();
                    s_type.Clear();
                    s_ammount.Clear();
                    FillSellGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Виберіть рядок");
            }
        }

        private void s_update_Click(object sender, EventArgs e)
        {
            if (s_id.Text != "" && s_date.Text != "" && s_name.Text != "" &&
s_type.Text != "" && s_ammount.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Збут set Date = @Date,Model =
@Model,Type = @Type, Amount= @Amount where ID = '" + s_id.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Date",
s_date.Text.ToString());
                    cmd.Parameters.AddWithValue("@Model",
s_name.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
s_type.Text.ToString());
                    cmd.Parameters.AddWithValue("@Amount",
s_ammount.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис оновлено");
                    s_id.Clear();
                    s_date.Clear();
                    s_name.Clear();
                    s_type.Clear();
                    s_ammount.Clear();
                    FillSellGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповніть всі поля");
            }
        }
```

```
        void FillSellGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            sell_list.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Збут", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                sell_list.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString());
            }
            dr.Close();
            conn.Close();
        }

        private void sell_list_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            DataGridViewRow row = this.sell_list.Rows[e.RowIndex];
            s_id.Text = row.Cells[0].Value.ToString();
            s_date.Text = row.Cells[1].Value.ToString();
            s_name.Text = row.Cells[2].Value.ToString();
            s_type.Text = row.Cells[3].Value.ToString();
            s_ammount.Text = row.Cells[4].Value.ToString();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Product product = new Product();
            product.Show();
            this.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            exportgridtopdf(paid_dataGridView1, "Звіт по оплаченм
Замовленням");
        }

        private void button2_Click(object sender, EventArgs e)
        {
            exportgridtopdf(unp_dataGridView, "Звіт по неоплаченм
Замовленням");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            exportgridtopdf(itemlist, "Звітність по Постачальникам");
```

```
        }

        private void button7_Click(object sender, EventArgs e)
        {
            exportgridtopdf(sell_list, "Звіт по Збуту");
        }

        private void button6_Click(object sender, EventArgs e)
        {
            Orders ord = new Orders();
            ord.Show();
        }

        private void Suppliers_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                ReleaseCapture();
                SendMessage(Handle, WM_NCLBUTTONDOWN, HT_CAPTION, 0);
            }
        }
    }
```

Клас/форма Orders.cs:

```
public partial class Orders : Form
    {
        public Orders()
        {
            InitializeComponent();
        }
        private void Orders_Load(object sender, EventArgs e)
        {
            FillComboBox();
        }

        void FillComboBox()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            dataGridView1.Rows.Clear();
            SqlCommand cmd = new SqlCommand("select * from Замовлення",
conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                dataGridView1.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString(), dr[6].ToString(),
dr[7].ToString(), dr[8].ToString());
            }
            dr.Close();
            conn.Close();
        }
```

```
        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
            {
                DataGridViewRow row = this.dataGridView1.Rows[e.RowIndex];
                textBox1.Text = row.Cells[0].Value.ToString();
                textBox2.Text = row.Cells[1].Value.ToString();
                textBox3.Text = row.Cells[2].Value.ToString();
                textBox4.Text = row.Cells[3].Value.ToString();
                textBox5.Text = row.Cells[4].Value.ToString();
                textBox6.Text = row.Cells[5].Value.ToString();
                textBox7.Text = row.Cells[6].Value.ToString();
                textBox8.Text = row.Cells[7].Value.ToString();
            }
        }

        private void close_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text
!= "" && textBox4.Text != "" && textBox5.Text != "" &&
                textBox6.Text != "" && textBox7.Text != "" && textBox8.Text
!= "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "update Замовлення set DStart = @DStart,
DEnd = @DEnd, Supplier = @Supplier, Model = @Model, Type = @Type, Price =
@Price, Ammount = @Ammount where ID = '" + textBox1.Text + "' ";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    double Price = double.Parse(textBox7.Text.ToString());
                    cmd.Parameters.AddWithValue("@DStart", textBox2.Text);
                    cmd.Parameters.AddWithValue("@DEnd", textBox3.Text);
                    cmd.Parameters.AddWithValue("@Supplier",
textBox4.Text.ToString());
                    cmd.Parameters.AddWithValue("@Model",
textBox5.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
textBox6.Text.ToString());
                    cmd.Parameters.AddWithValue("@Price", Price);
                    cmd.Parameters.AddWithValue("@Ammount",
textBox8.Text.ToString());
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис оновлено");
                    textBox1.Clear();
                    textBox2.Clear();
                    textBox3.Clear();
                    textBox4.Clear();
```

```
                    textBox5.Clear();
                    textBox6.Clear();
                    textBox7.Clear();
                    textBox8.Clear();
                    FillUpdateGridView();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Оберіть запис");
            }

        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text
!= "" && textBox4.Text != "" && textBox5.Text != "" &&
                textBox6.Text != "" && textBox8.Text != "")
            {
                try
                {
                    SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
                    string query = "INSERT INTO Замовлення(DateStart,
DateEnd, Supplier, Model, Type, Price, Ammount, Supply) VALUES(@DateStart,
@DateEnd, @Supplier, @Model, @Type, @Price, @Ammount, @Supply)";
                    SqlCommand cmd = new SqlCommand(query, conn);
                    double Price = double.Parse(textBox7.Text.ToString());
                    cmd.Parameters.AddWithValue("@DateStart",
textBox2.Text.ToString());
                    cmd.Parameters.AddWithValue("@DateEnd",
textBox3.Text.ToString());
                    cmd.Parameters.AddWithValue("@Supplier",
textBox4.Text.ToString());
                    cmd.Parameters.AddWithValue("@Model",
textBox5.Text.ToString());
                    cmd.Parameters.AddWithValue("@Type",
textBox6.Text.ToString());
                    cmd.Parameters.AddWithValue("@Price", Price);
                    cmd.Parameters.AddWithValue("@Ammount",
textBox8.Text.ToString());
                    cmd.Parameters.AddWithValue("@Supply", "-");
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("Запис додано");
                    textBox1.Clear();
                    textBox2.Clear();
                    textBox3.Clear();
                    textBox4.Clear();
                    textBox5.Clear();
                    textBox6.Clear();
```

```
                    textBox7.Clear();
                    textBox8.Clear();
                    FillComboBox();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                }
            }
            else
            {
                MessageBox.Show("Заповність всі поля");
            }
        }

        void FillUpdateGridView()
        {
            SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" + Application.StartupPath +
@"\MyBD.mdf;Integrated Security=True;Connect Timeout=30");
            dataGridView1.Rows.Clear();
            SqlCommand cmd = new SqlCommand("SELECT * FROM Замовлення",
conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                dataGridView1.Rows.Add(dr[0].ToString(), dr[1].ToString(),
dr[2].ToString(), dr[3].ToString(),
                    dr[4].ToString(), dr[5].ToString(), dr[6].ToString(),
dr[7].ToString());
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            exportgridtopdf(dataGridView1, "Звіт по всім Замовленням");
        }

        public void exportgridtopdf(DataGridView dgw, string filename)
        {
            BaseFont bf = BaseFont.CreateFont(Application.StartupPath +
"\\Arial.ttf", BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
            PdfPTable pdftable = new PdfPTable(dgw.Columns.Count);
            pdftable.DefaultCell.Padding = 3;
            pdftable.WidthPercentage = 100;
            pdftable.HorizontalAlignment = Element.ALIGN_LEFT;
            pdftable.DefaultCell.BorderWidth = 1;

            iTextSharp.text.Font text = new iTextSharp.text.Font(bf,
iTextSharp.text.Font.DEFAULTSIZE, iTextSharp.text.Font.NORMAL);
            foreach (DataGridViewColumn column in dgw.Columns)
            {
                PdfPCell cell = new PdfPCell(new Phrase(column.HeaderText,
text));
                cell.BackgroundColor = new iTextSharp.text.BaseColor(240,
240, 240);
                pdftable.AddCell(cell);
```

```
            }

            foreach (DataGridViewRow row in dgw.Rows)
            {
                foreach (DataGridViewCell cell in row.Cells)
                {
                    pdftable.AddCell(new Phrase(cell.Value.ToString(),
text));
                }
            }

            var savefiledialoge = new SaveFileDialog();
            savefiledialoge.FileName = filename;
            savefiledialoge.DefaultExt = ".pdf";
            if (savefiledialoge.ShowDialog() == DialogResult.OK)
            {
                using (FileStream stream = new
FileStream(savefiledialoge.FileName, FileMode.Create))
                {
                    Document pdfdoc = new Document(PageSize.A4, 10f, 10f,
10f, 0f);
                    PdfWriter.GetInstance(pdfdoc, stream);
                    pdfdoc.Open();
                    pdfdoc.Add(pdftable);
                    pdfdoc.Close();
                    stream.Close();
                }
            }
        }
    }
```

ДОДАТОК В

(Обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

Дипломна робота

на тему: «Система автоматизації обліку матеріальних цінностей на складі»

Керівник:
канд. пед. наук, доцент
Праворська Н.І.

Виконав:
студент IV курсу
групи ІПЗ-18-1
Лучицький О.Ю.

Актуальність дипломного проекту:

Складський облік лежить в основі практично будь-якої підприємницької діяльності. Успіх діяльності підприємства завжди залежав не тільки від якості продукції, але й від можливості керувати внутрішніми процесами.

Актуальність теми пов'язана з тим, що більша частина часу працівників складського відділу старого зразка в основному напрямлена на оформлення різноманітної документованої звітності. Використання комп'ютерних технологій забезпечує менш затратну у часі та більш ефективну діяльність.

Мета проекту:

Метою проекту є розробка програмного забезпечення для обліку матеріальних цінностей, яка надає доступ до бази даних магазину комп'ютерної техніки та дозволяє виконання операцій редагування бази через засоби інтерфейсу.

Інтерфейс даного програмного забезпечення має дозволяти слідкувати за наявністю товару методом взаємодії з табличною базою даних, вести запис постачальників товару, відстежувати вчасне прибуття товару від постачальника на склад та збут товару зі складу.

Завдання проекту:

Для досягнення мети, для дипломного проекту були визначені наступні завдання:

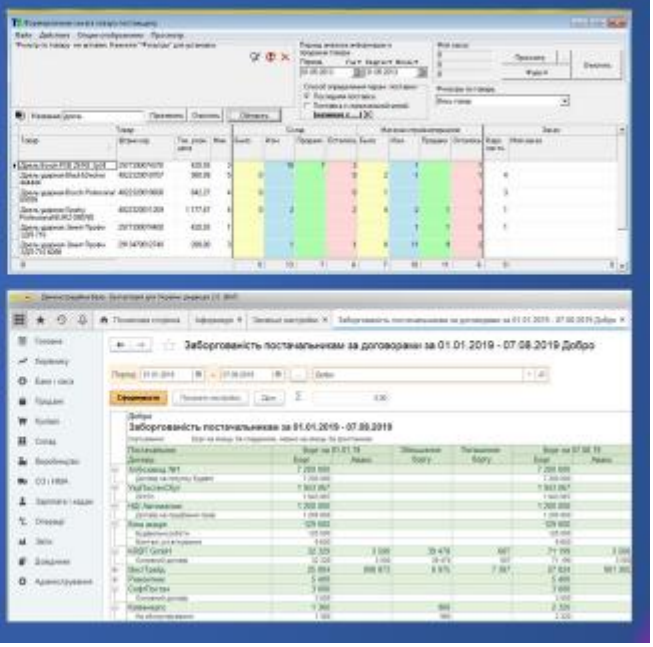
- дослідити предметну область обліку матеріальних цінностей під час процесу роботи складського відділу;
- проаналізувати існуючі рішення в досліджуваній області та визначити на їх основі вимоги до розроблюваного ПЗ;
- розробити архітектуру програмного забезпечення та базу даних;
- вибрати технології для подальшої розробки;
- розробити та протестувати програмне забезпечення на відповідність вимогам та наявність недоліків.

Аналіз існуючих рішень:

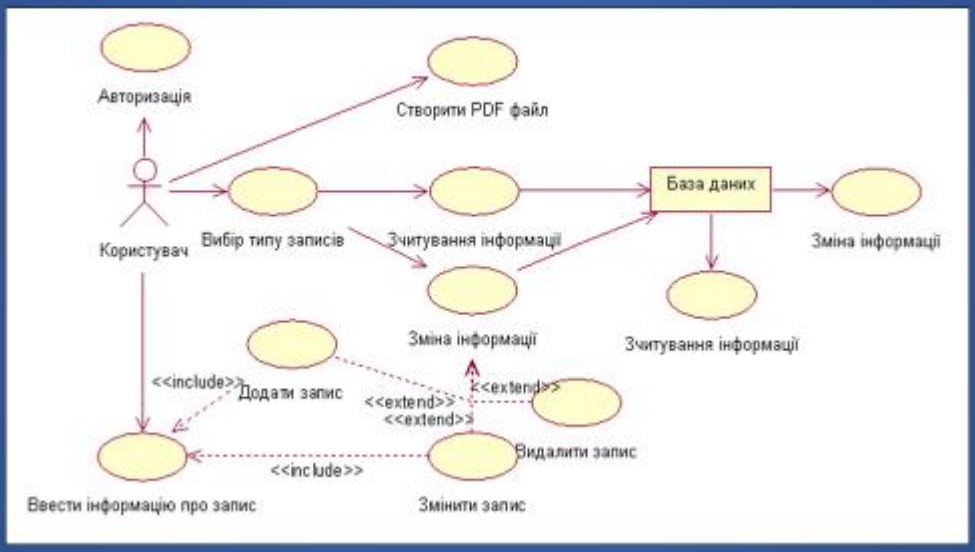
Найпопулярніші рішення в області складського обліку представлені комплексними продуктами які, здебільшого зачіпляють всі області обліку. Такі продукти бувають дорогими (особливо з тимчасовою ліцензією), та дещо «перевантаженими» для малих та менших середніх підприємств. Що не менш важливо, деякі з таких продуктів потребують знань в області бухгалтерського обліку.

Розглянуті рішення:

- **ТоргСофт**
- **BAS Бухгалтерія**



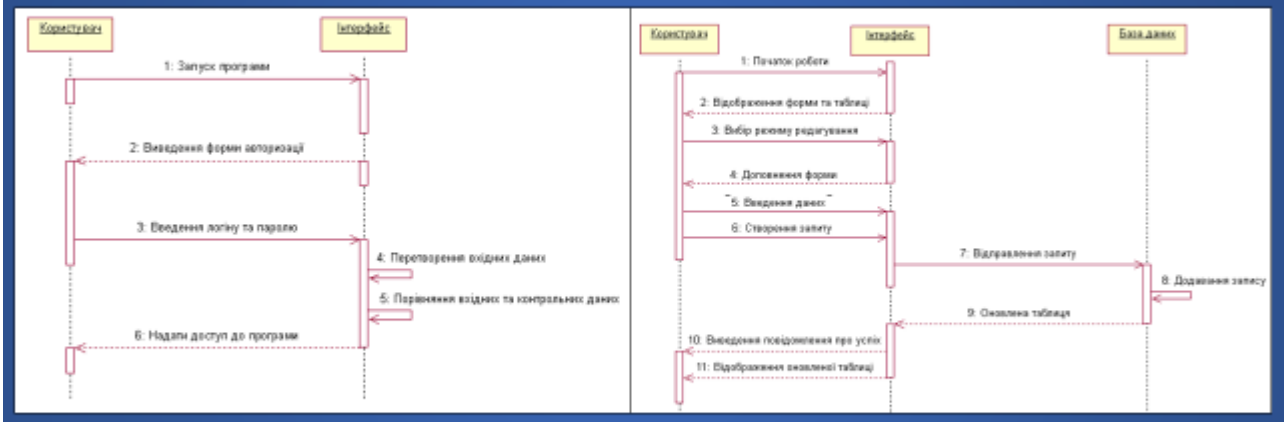
Діаграма варіантів використання:



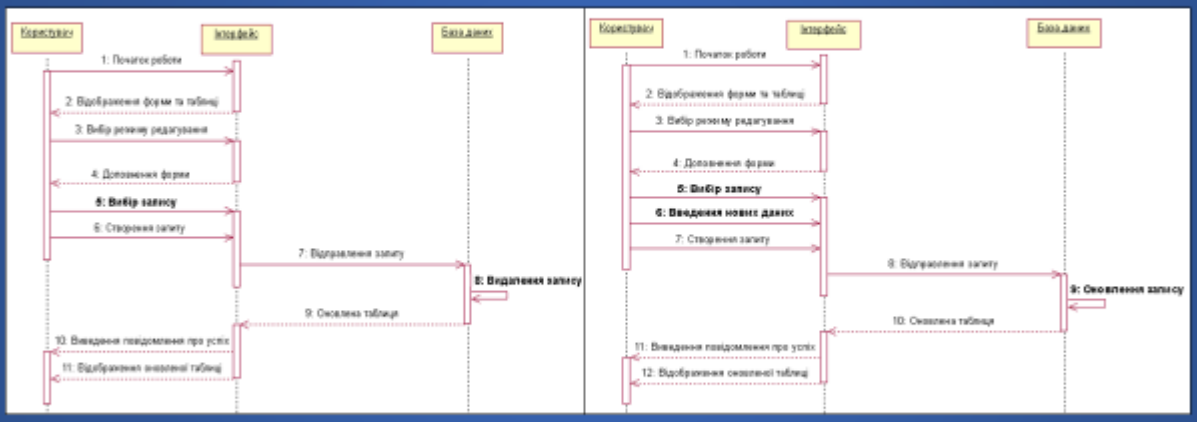
Архітектура ПЗ:

Розроблюваний додаток має просту архітектуру:

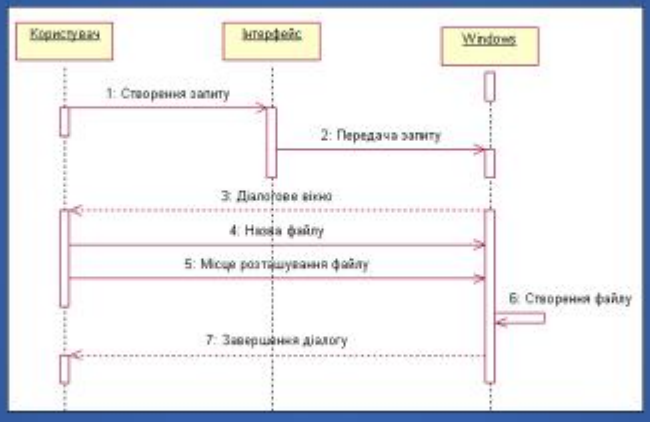
Діаграми послідовностей:

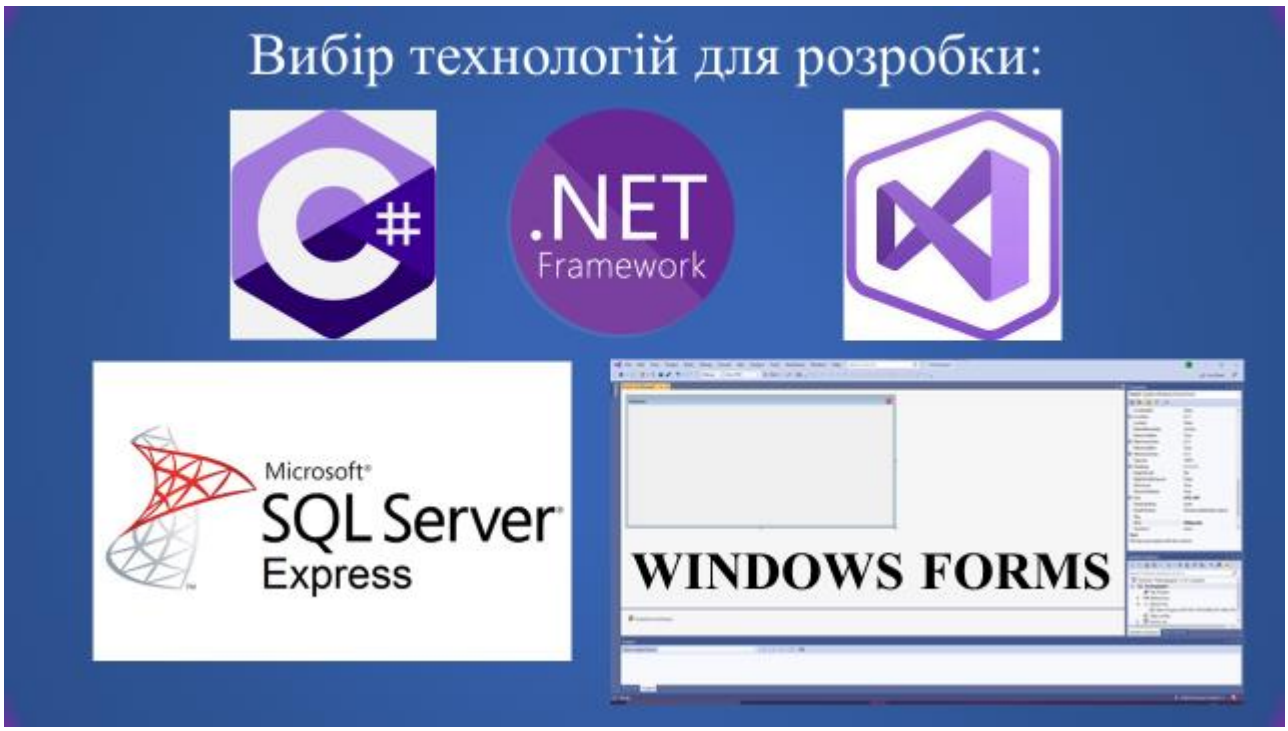


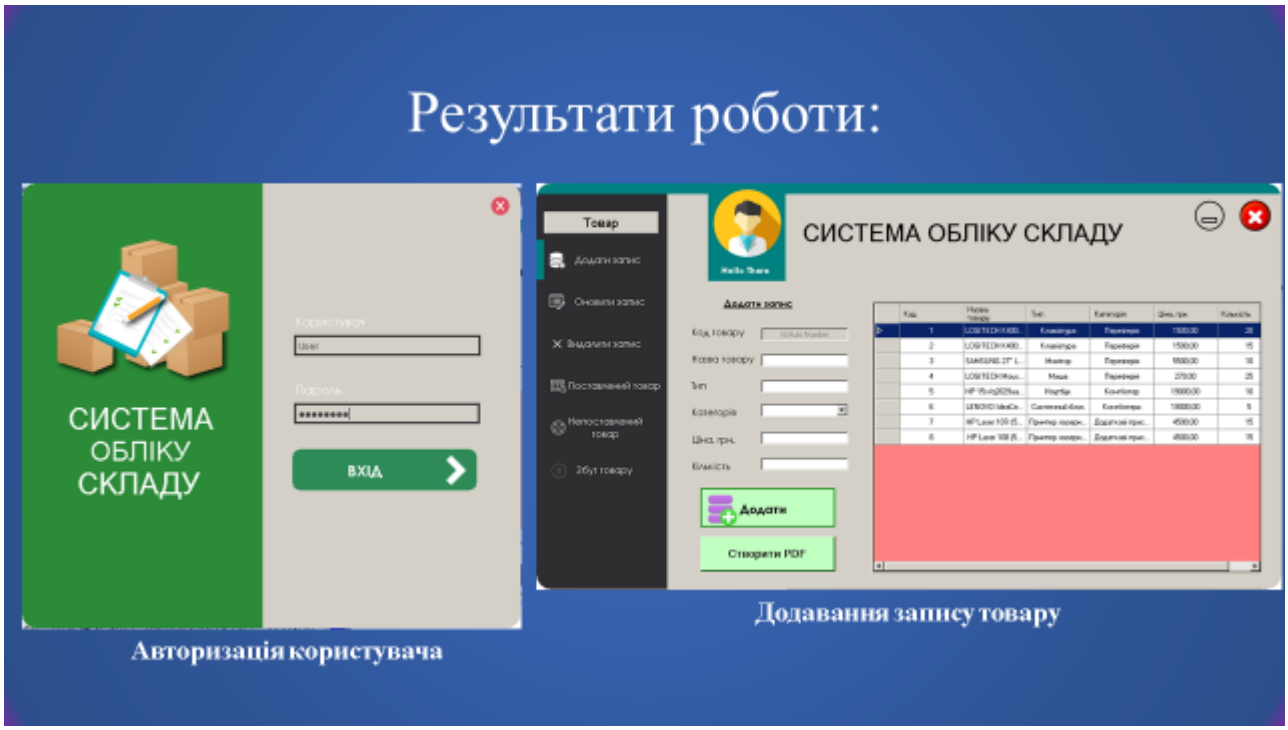
Діаграми послідовностей:

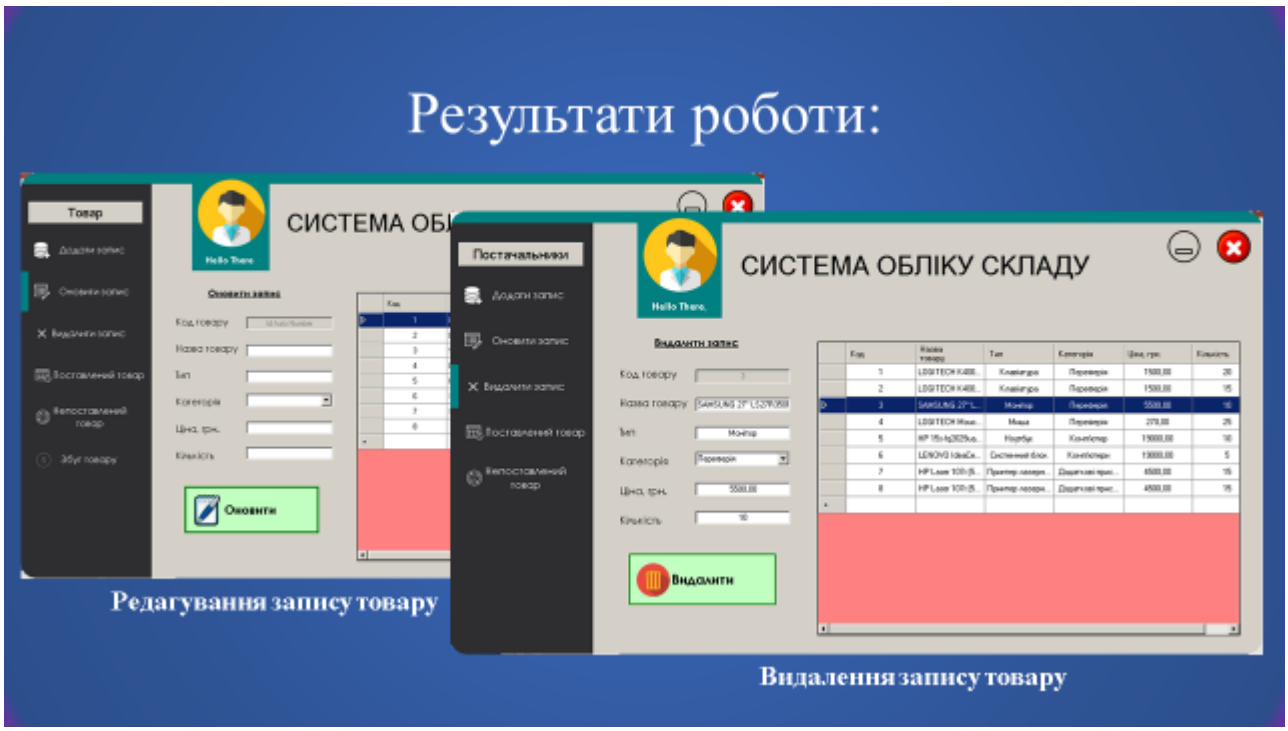


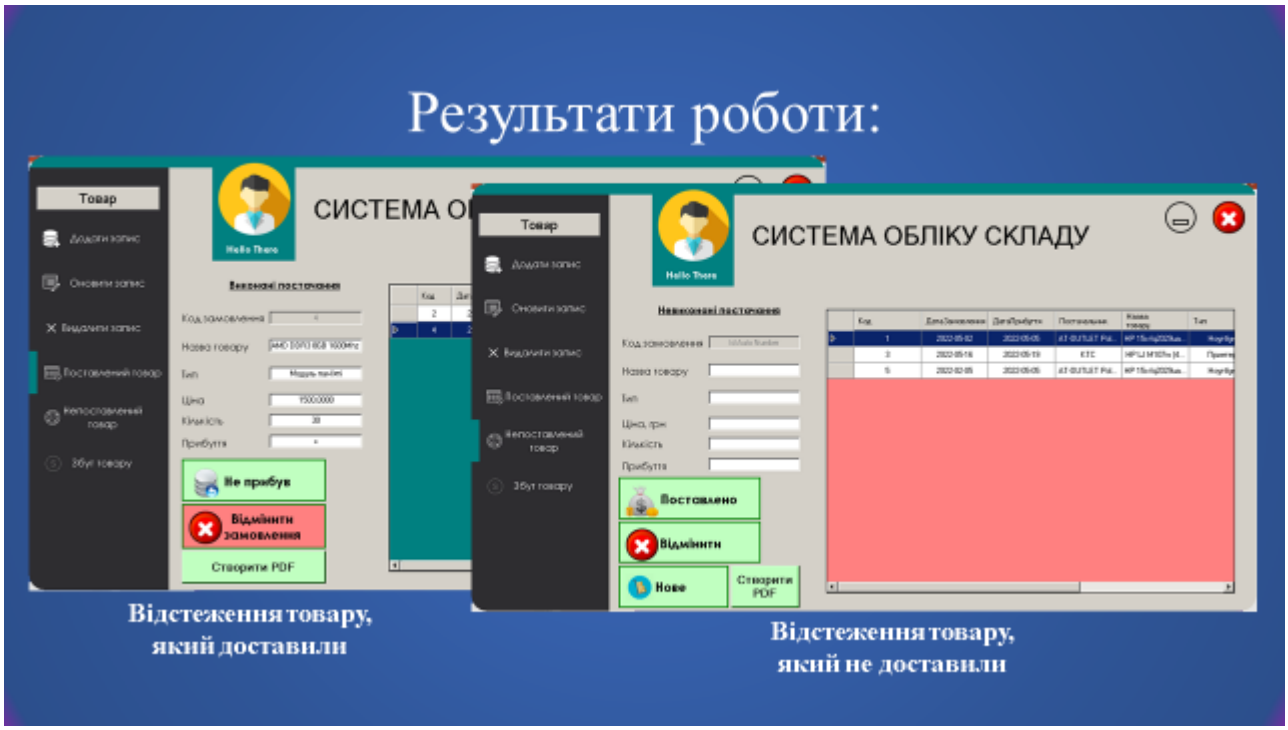
Діаграми послідовностей:



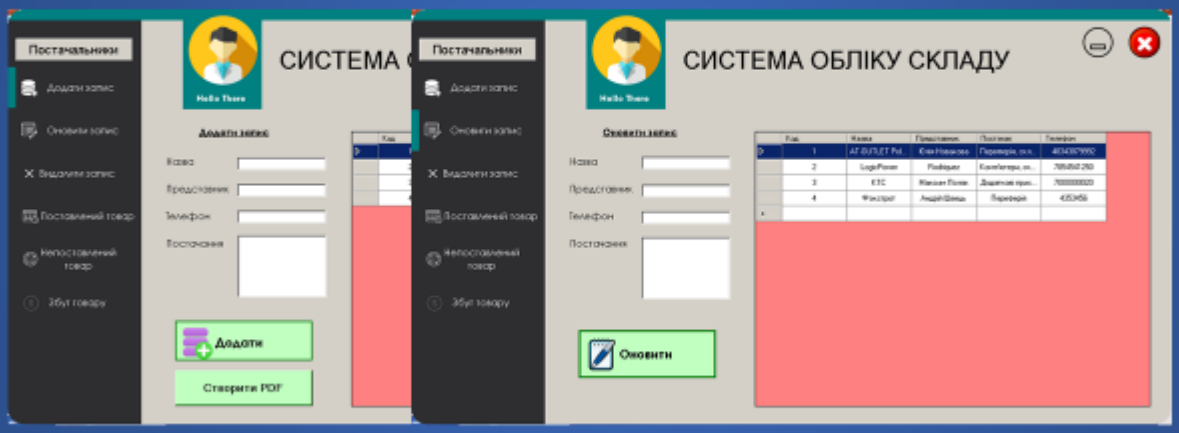

Вибір технологій для розробки:
.NET
Framework
Microsoft
SQL Server
Express
WINDOWS FORMS


Результати роботи:
СИСТЕМА
ОБЛІКУ
СКЛАДУ
ВХІД
Авторизація користувача
СИСТЕМА ОБЛІКУ СКЛАДУ
Товар
Додати запис
Оновити запис
Видалити запис
Додати
Створити PDF
Додавання запису товару


Результати роботи:
Товар
Додати запис
Оновити запис
Видалити запис
Поставлений товар
Непоставлений товар
Збут товару
Hello There
Оновити
Редагування запису товару
Постачальники
СИСТЕМА ОБЛІКУ СКЛАДУ
Видалити запис
Видалити
Видалення запису товару


Результати роботи:
Товар
Додати запис
Оновити запис
Видалити запис
Поставлений товар
Непоставлений товар
Збут товару
Hello There
Не прибув
Відмінити замовлення
Створити PDF
Відстеження товару, який доставили
СИСТЕМА ОБЛІКУ СКЛАДУ
Поставлено
Відмінити
Нове
Створити PDF
Відстеження товару, який не доставили

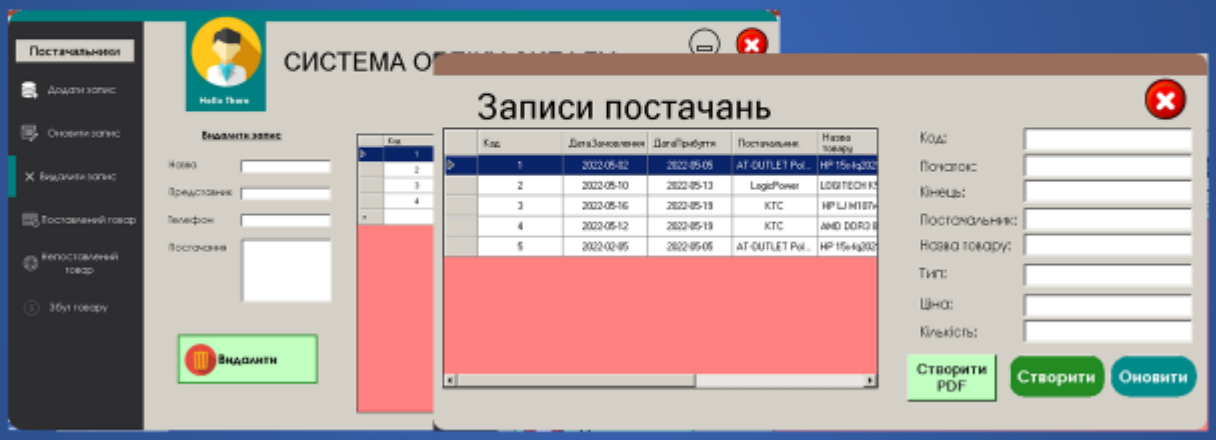
Результати роботи:



Додавання запису про постачальника

Оновлення запису про постачальника

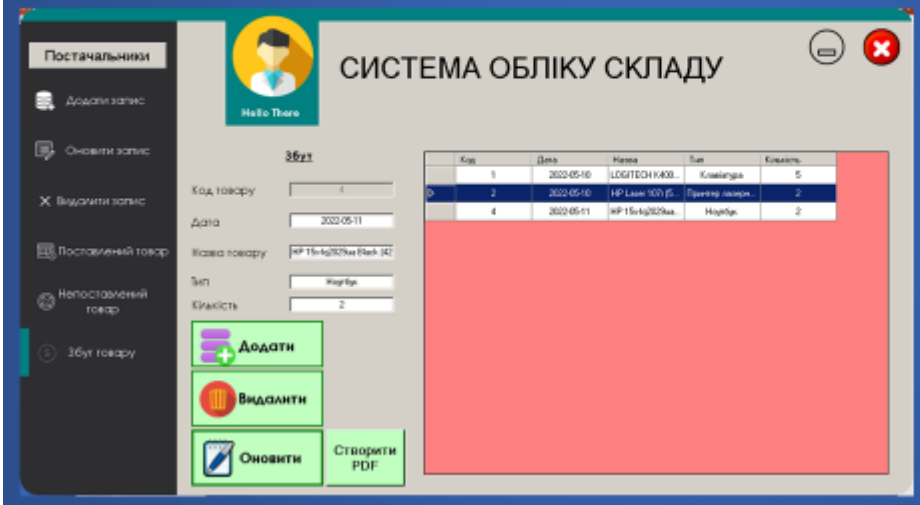
Результати роботи:



Видалення запису про постачальника

Додавання та оновлення записів про постачання

Результати роботи:



Перегляд та редагування записів про збут товару

Тестування ПЗ:

- Тестування функціональності:

1. модульне;
2. інтеграційне;
3. системне;

- Тестування проводилося протягом розробки.

Висновок:

Під-час створення дипломної роботи був аналіз існуючих рішень в області складського обліку з виявленням їх переваг, недоліків та функцій, а також розробки програмного забезпечення з інтуїтивно зрозумілим інтерфейсом.

Створене програмне забезпечення дозволяє використовувати електронну базу даних для ведення складського обліку. Окрім цього, програмний код дозволяє подальше оновлення функціоналу.

Дововідь завершено

Дякую за увагу!

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Лучицького О. Ю.
Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-18-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

30.05.2022р.
дата

О. Луч
підпис

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1011377402

Дата перевірки:
30.05.2022 14:36:00 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
30.05.2022 14:40:35 EEST

ID користувача:
100005589

Назва документа: **Диплом_Записка_Лучицький_О_Ю_ІПЗ_18_1!!!**

Кількість сторінок: **78** Кількість слів: **12564** Кількість символів: **102473** Розмір файлу: **1.79 MB** ID файлу: **1011261885**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

10.9%
Схожість

Найбільша схожість: **5.44%** з джерелом з Бібліотеки (ID файлу: **1008215667**)

4.57% Джерела з Інтернету	108	Сторінка 80
7.62% Джерела з Бібліотеки	134	Сторінка 81

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	14 сторінок

Mon May 30 13:40:30 EEST 2022, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 7.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 7%**

ID: 104219 Назва: Система автоматизації обліку матеріальних цінностей на складі Додано в БД: 2022-05-30 Автора: О. Ю. Лучицький Керівники: Н. І. Праворська Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	72667	1101	7018 (10%)	89 (8%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник Лучицький Олег Юрійович

Тема Система автоматизації обліку матеріальних цінностей на складі

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломної роботи:

Кількість листів креслень _____; кількість сторінок записки 128

1.Короткий зміст ДР та прийнятих рішень У дипломному проекті було досліджено і проаналізовано предметну область та визначеного усі вимоги до розроблюваного програмного забезпечення. Був проведений аналіз існуючих програм на ринку, розглянуто їх переваги і недоліки, та доведено актуальність розробки нового програмного забезпечення. Було розглянуто інструменти для реалізації дипломного проекту, в результаті чого створено програмне забезпечення. Також було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність ДР поставленому завданню Дипломна робота освітнього ступеня «бакалавр» у повній мірі відповідає поставленому завданню, як у теоретичній, так і практичній її частині, з дотриманням всіх вимог

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначена архітектура розроблюваного програмного забезпечення та бази даних, з якою воно працює. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. В четвертому розділі було виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми.

4. Позитивні сторони роботи Тематика дипломного проекту є актуальною, оскільки облік є основою діяльності будь-якого підприємства і використання програмних засобів для автоматизації роботів у цьому напрямі є складовою підприємства, що розвивається та намагається досягти кращих результатів своєї діяльності.

5. Негативні сторони роботи У проекті функціонал додатку пов'язаний з роботою з єдиною локальною базою даних без можливості використання інших. Також функціонал додатку не передбачає утворення звітності по параметрам, якщо такою не вважати можливість сортування записів бази даних засобами інтерфейсу.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконано відповідно до теми дипломної роботи з дотриманням вимог стандартів. У загальному графічне оформлення виконано на достатньому рівні. Пояснювальна записка відповідає вимогам стандартів до її оформлення.

7. Відгук про роботу в цілому В цілому дипломна робота заслуговує позитивної оцінки. Весь матеріал дипломної роботи структурований, чіткий та послідовний. Усі розділи роботи є послідовними та логічними, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики дипломної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для вирішення поставленої задачі.

8. Інші зауваження

9. Оцінка дипломної роботи Розглянувши позитивні і негативні сторони представленої дипломної роботи, можна зробити висновок, що вона заслуговує оцінки «відмінно»

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Говорущенко Т.О., д.т.н., проф., зав. каф. КІІС ХНУ

"30" травня 2022 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Система автоматизації обліку матеріальних цінностей на складі»

Автор: Лучицький Олег Юрійович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Праворська Наталія Іванівна, канд. пед. наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломного проекту системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання на проектування, відомість документів, у структурі ЗМІСТУ, написах в рамках, назвах розділів/підрозділів тощо) та в назвах переліку джерел посилання;

2) в якості запозичень системою зафіксовано деякі послідовності вихідного коду, які є стандартними мовними конструкціями, спільними для великої кількості задач, і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформлені посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності схожості, складає 10,9% і адресується до 108 першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ Н. І. Праворська

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк