ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва мовою C++»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ілля ФІЛЬ
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Ілля ФІЛЬ

Керівник: Ірина ЩЕРБИНА
к.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Філь Іллі Владиславовичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва мовою С++»
керівник кваліфікаційної роботи к.т.н., доц., доцент кафедри ІПЗ Ірина ЩЕРБИНА, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоритичні відомості про методи захисту ліків від підробки та незаконного виробництва.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз існуючих існуючих методів та технологій захисту ліків від підробки та незаконного виробництва

2. Проектування застосунку для захисту ліків від підробки та незаконного виробництва

3. Програмна реалізація та опис функціонування застосунку для захисту ліків від підробки та незаконного виробництва

4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма класів.
6. Діаграма станів
7. Схема бази даних
8. Екранні форми
9. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Вибір інструментів та технологій для розробки застосунку	14.03-20.03.2024	
4	Проектування застосунку для захисту ліків від підробки та незаконного виробництва	21.03-29.03.2024	
5	Програмна реалізація застосунку	30.03-08.04.2024	
6	Тестування застосунку	09.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Ілля ФІЛЬ
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЩЕРБИНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 15 рис., 12 джерел.

Мета роботи – покращення процесу захисту ліків від підробки та незаконного виробництва за допомогою програмного забезпечення мережевої системи DrugGuard мовою C++

Об'єкт дослідження – процес захисту від підробки та незаконного виробництва ліків.

Предмет дослідження – програмне забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва.

В роботі проаналізовано наявні проблеми системи захисту ліків від підробки та незаконного виробництва, які є загрозою для здоров'я населення та економіки. Досліджено кращі практики зарубіжного досвіду у сфері захисту ліків та визначено функціональні вимоги до нової системи. Розроблено програмне забезпечення мережевої системи, яка включає авторизацію, реєстрацію лікарських препаратів, відстеження переміщення ліків, генерації звітів, програмно реалізовані ключові функціональні можливості, зокрема, зберігання та обробки даних про виробництво, переміщення та продаж лікарських засобів. Проведено тестування системи на відповідність вимогам безпеки, масштабованості та продуктивності. В роботі використано мову програмування C++, середовище розробки Visual Studio та SQL Management Studio для роботи з базами даних.

КЛЮЧОВІ СЛОВА: ЗАХИСТ ЛІКІВ, ПІДРОБКА, C++, VISUAL STUDIO, SQL MANAGEMENT STUDIO.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

SQL - Structured Query Language

TCP/IP - Transmission Control Protocol/Internet Protocol

UDP (User Datagram Protocol)

RFID (Radio Frequency Identification)

ВСТУП

Актуальність теми. Актуальність теми дипломної роботи "Розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва мовою C++" важко переоцінити, оскільки проблема фальсифікації ліків є однією з найсерйозніших загроз сучасного охорони здоров'я. Підроблені медикаменти не лише неефективні, але й можуть бути небезпечними для здоров'я та життя пацієнтів. Вони стають причиною сотень тисяч смертей щороку, особливо в країнах з низьким та середнім рівнем доходу, де контроль за якістю лікарських засобів є слабшим.

Розробка програмного забезпечення для захисту ліків від підробки та незаконного виробництва є актуальною також з економічної точки зору. Підробка ліків завдає значних фінансових збитків фармацевтичним компаніям, підриває довіру до медичних установ та спричиняє значні витрати на лікування ускладнень, викликаних неякісними медикаментами. Використання мови програмування C++ у цьому контексті є обґрунтованим вибором, оскільки вона забезпечує високу продуктивність, ефективність роботи з мережевими протоколами та можливість розробки складних систем захисту даних.

Мережеві системи для захисту ліків можуть включати в себе різні технології, такі як блокчейн для відстеження походження та ланцюга поставок медикаментів, криптографічні методи для захисту інформації та системи аутентифікації для перевірки дійсності ліків. Використання C++ дозволяє створювати високооптимізовані програми, які можуть працювати у реальному часі та обробляти великі обсяги даних, що є критично важливим для ефективного функціонування такої системи.

Зважаючи на глобалізацію ринків та зростаючі обсяги міжнародної торгівлі медикаментами, створення ефективних програмних рішень для захисту від підробок є вкрай актуальним. Це не лише сприяє підвищенню безпеки пацієнтів, але й допомагає зміцнити загальну систему охорони здоров'я, підвищуючи її стійкість до загроз, пов'язаних із підробкою ліків. Тому розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва є важливим та актуальним напрямком досліджень і розробок.

Об'єкт дослідження - Процес захисту від підробки та незаконного виробництва ліків.

Предмет дослідження - Програмне забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва.

Мета дослідження - Покращення процесу захисту ліків від підробки та незаконного виробництва за допомогою програмного забезпечення мережевої системи DrugGuard мовою C++

Відповідно до мети було поставлено наступні **завдання:**

- Дослідити проблему захисту ліків від підробки
- Визначити основні мережеві протоколи транспортного рівня
- Проаналізувати мову програмування C++
- Проаналізувати середовища розробки C++
- Розробити програмне рішення
- Проаналізувати розроблене програмне рішення

Структура роботи. Робота складається з трьох розділів, шести підрозділів, висновків та списку використаних джерел. Загальний обсяг роботи - 60 сторінок.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема захисту ліків від підробки

Виробникам і власникам торгових марок доступна велика кількість технологій боротьби з підробками, починаючи від дуже простих, але ефективних і закінчуючи надзвичайно складними та надзвичайно безпечними. Більшість може бути реалізовано на одному чи кількох компонентах упаковки, але деякі функції можна застосувати навіть на рівні продукту шляхом прямого маркування або використання фізичних чи хімічних маркерів у рецептурі.

Метою функції захисту від підробок є насамперед уможливлення автентифікації продукту дослідниками галузі або, в ідеалі, широким загалом. Друга функція може полягати в тому, щоб діяти як стримуючий фактор для будь-кого, хто розглядає підробку продукту на основі труднощів або витрат, пов’язаних із ймовірністю виявлення та, отже, судового переслідування. Необхідно підкреслити, що захисні пристрої на компонентах упаковки не забезпечують автентичності вмісту, який може бути замінений або підроблений. Пристрої безпеки самі по собі не зменшують кількість підробок, але розроблені, щоб полегшити їх виявлення.

Технології боротьби з підробками можна узагальнено класифікувати таким чином:

- Явні або видимі особливості
- Приховані, або приховані маркери
- Криміналістична техніка
- Серіалізація/відстеження та трасування

У цьому документі буде розглянуто кожну з цих 4 категорій, уникаючи конкретних посилань на будь-який ліцензований продукт або постачальника. Однак слід визнати, що деякі з цих технологій захищені міжнародними патентами та можуть бути доступні лише від ліцензованих постачальників за умови відповідних роялті або ліцензійних зборів. З іншого боку, деякі з них можна застосувати власними силами, з невеликими витратами на матеріали та зусилля, а більшість можна придбати у постачальників із хорошою репутацією, деякі з яких спеціалізуються на застосуванні безпеки.

АНТИПІДРОБНІ ТЕХНОЛОГІЇ

ОКРЕТНІ (видимі) функції

Відкриті функції призначені для того, щоб дозволити кінцевим користувачам перевірити автентичність пачки. Такі елементи, як правило, добре видно, і їх важко або дорого відтворити. Слід зазначити, що відкриті функції можуть збільшити значні витрати, можуть обмежити доступність поставок і вимагають навчання кінцевих користувачів, щоб бути ефективними. Якщо використовуються явні функції, фальшивомонетники часто використовують просту копію, яка імітує справжній пристрій, достатньо добре, щоб заплутати пересічного користувача.

Вони також вимагають максимальної безпеки в процедурах постачання, обробки та утилізації, щоб уникнути несанкціонованого перенаправлення. Їх слід застосовувати таким чином, щоб їх неможливо було повторно використати або видалити, не зіпсувавши вигляд і не спричинивши пошкодження упаковки – інакше оригінальні використані компоненти можуть бути перероблені з підробленим вмістом, створюючи хибне враження автентичності. З цієї причини відкритий пристрій може бути включений у функцію Tamper Evident для додаткової безпеки.

Мабуть, найвідомішою відкритою особливістю є голограма «голуб», яка використовується для захисту кредитних карток протягом багатьох років. Зазвичай голограма містить зображення з деякою ілюзією тривимірної конструкції або видимої глибини та особливого розділення.

Голограми та подібні оптично змінні пристрої (OVD) можна зробити більш ефективними, якщо включити їх у функцію захисту від несанкціонованого доступу або як невід'ємну частину первинної упаковки (наприклад, блістерної фольги). Вони можуть бути включені в розривні смуги в обгорткових плівках або у вигляді ниток, вбудованих у паперові підкладки.

Однак деякі голограмні мітки легко та досвідчено скопійовані або змодельовані, і вони часто можуть покладатися на приховані приховані елементи для автентифікації.

Оптично змінні пристрої (OVD)

OVD також включає широкий спектр альтернативних пристроїв, схожих на голограми, але часто без будь-якого 3D-компонента. Як правило, вони передбачають перевертання зображення або переходи, часто включаючи перетворення кольорів або монохроматичні контрасти.

Як і голограми, вони, як правило, складаються з прозорої плівки, яка служить носієм зображення, а також відбиваючого шару підкладки, який зазвичай є дуже тонким шаром алюмінію. Інші метали, такі як мідь, можуть використовуватися для надання характерного відтінку для спеціалізованих програм безпеки.

Додатковий захист може бути забезпечений процесом часткової деметалізації, під час якого частина світловідбиваючого шару видаляється хімічним шляхом, щоб надати складного контуру зображенню, як це можна побачити на багатьох банкнотах. В якості альтернативи відбиваючий шар може бути таким тонким, щоб бути прозорим, у результаті чого утворюється прозора

плівка з більшою кількістю примарного відбиваючого зображення, видимого під певними кутами огляду та освітлення. Часткове видалення металевого шару є більш обмеженим процесом, що підвищує як рівень безпеки, так і вартість.

Захисні фарби та плівки, що змінюють колір

Вони можуть відображати позитивні зміни в кольорі відповідно до кута огляду та можуть бути ефективними або як явний графічний елемент, або шляхом включення в захисну печатку.

Пігменти, що змінюють колір, — це дрібно подрібнені металеві ламінати, які потрібно покласти в товсту непрозору плівку для досягнення оптичного ефекту, і тому вони краще підходять для таких методів друку, як глибокий і трафаретний друк, а не для літографічного друку. Їх безпека полягає в специфічності та динаміці зміни кольору (наприклад, від синього до золотого), у поєднанні з труднощами та витратами, пов’язаними з виготовленням. Вони доступні лише від обмеженої кількості постачальників пігменту через кілька спеціалізованих виробників чорнила. Позитивна автентифікація може включати судово-медичне (мікроскопічне) дослідження та вбудовані мітки.

Плівки, що змінюють колір, використовувалися для забезпечення безпеки, включаючи багатошарове нанесення тонких плівок для створення структури з унікальними дифракційними властивостями та яскравими переходами кольорів. Вони можуть бути нанесені як захисні пломби або етикетки, що захищають від несправностей.

Графіка безпеки

Кольоровий друк із тонкими лініями, схожий на друк банкнот, що включає низку явних і прихованих елементів дизайну, таких як гільйоші, лінійна модуляція та лінійне тиснення. Їх можна використовувати як фон у окремій зоні, такій як область накладення, або як повну упаковку графіки, і їх можна друкувати за допомогою звичайної офсетної літографії або для підвищення безпеки за

допомогою глибокого друку. Помітне використання пастельних «точкових» кольорів ускладнює сканування та відтворення дизайну, а безпека додатково підвищується за рахунок включення ряду прихованих елементів дизайну, таких як мікротекст і приховані зображення.

Порядкова нумерація товару

Унікальна послідовна нумерація кожної упаковки або етикетки в партії може полегшити виявлення підробок у ланцюжку постачання. Якщо надруковано видимо, воно забезпечує напіввідкритий засіб автентифікації за допомогою посилання на захищену базу даних, оскільки дублікати або недійсні номери будуть відхилені. Основні недоліки послідовної нумерації полягають у тому, що послідовність передбачувана і легко відтворюється, а кінцевим користувачам потрібні певні засоби доступу до бази даних. Більш безпечним варіантом є серіалізація за допомогою псевдовипадкової послідовності, що не повторюється, і це обговорюється в розділі Track and Trace (4).

Маркування на виробі

Технології маркування на продукті дозволяють розміщувати спеціальні зображення або коди на звичайних пероральних лікарських формах. Ці відкриті технології може бути важко відтворити та запропонувати технологію безпеки на рівні таблетки. Цей додатковий рівень безпеки ефективний, навіть якщо продукти відокремлені від оригінальної упаковки.

Відкриті функції представляють собою спробу передати автентифікацію в руки широкої громадськості. Однак, щоб бути ефективними, вони вимагають освіти та обізнаності громадськості, що особливо складно на ринках, що розвиваються з найбільшими труднощами. Слід також зазначити, що чим ширше використовується одна відкрита захисна технологія, тим привабливіше для фальшивомонетників її перемогти.

ПРИХОВАНІ (Приховані) функції

Мета прихованої функції — дозволити власнику бренду ідентифікувати підроблений продукт. Широка громадськість не знатиме про його присутність і не матиме засобів перевірити це. Приховану функцію не повинно бути легко виявити або скопіювати без спеціальних знань, і її деталі потрібно контролювати на основі «необхідності знати». Якщо скомпрометовано або оприлюднено, більшість прихованих функцій втратять частину, якщо не всю цінність безпеки. З цієї причини такі методи не будуть детально розкриватися в цій статті.

Приклади:

Невидимий друк

За допомогою спеціальних чорнил можна надрукувати невидимі позначки майже на будь-якій основі, які з’являються лише за певних умов, наприклад через УФ- чи ІЧ-освітлення. Вони можуть бути сформульовані так, щоб відображати різні кольори з освітленням на різних довжинах хвиль.

Вбудоване зображення

Невидиме зображення може бути вбудовано в графіку пакета, яку можна переглядати лише за допомогою спеціального фільтра, і її неможливо відтворити за допомогою звичайних засобів сканування. Наслідки можуть бути досить драматичними, але все ж добре прихованими.

Цифрові водяні знаки

Невидимі дані можна цифрово закодувати в графічних елементах і перевірити за допомогою зчитувача та спеціального програмного забезпечення. Дані можуть бути отримані за допомогою веб-камери, мобільного телефону чи іншого обладнання для сканування, але цифрова інформація невидима людським оком, і спроби відтворити її будуть виявлені через погіршення якості вбудованих даних.

Приховані позначки та друк

Спеціальні позначки та друк можуть бути нанесені таким чином, що не привертають уваги та нелегко скопіювати. Їхня ефективність залежить від поєднання секретності та тонкості, тому тут не обговорюватимуться додаткові подробиці.

Дизайн із захистом від копіювання або сканування

Фонові візерунки тонких ліній виглядають як рівномірні тони, але під час сканування або копіювання виявляють приховане зображення, яке раніше не було видимим. Зазвичай використовуються на захищених документах, щоб запобігти фотокопіюванню, вони можуть застосовуватися до упаковки продукту як відтінок фону.

Лазерне кодування

Застосування пакетних змінних деталей за допомогою лазерного кодування вимагає спеціального та дорогого обладнання, що призводить до розпізнаваних артефактів, які може бути важко імітувати. Лазерні коди можна наносити на коробки та етикетки, а також на пластикові та металеві компоненти.

Підкладки

Існує багато способів введення прихованих маркерів у підкладку, таких як видимі або ультрафіолетові флуоресціючі волокна або хімічні реагенти на картоні чи папері. Водяні знаки можуть бути вставлені в папір для листівок або металеві нитки, вплетені в основний матеріал, можливо, з явною ознакою OVD. Вони вимагають спеціального джерела постачання та великого обсягу виробництва, що, якщо це доступно, призводить до дуже ефективного варіанту.

Запах

Мікроінкапсульовані характерні запахи можна застосовувати як добавку до чорнила або покриття, щоб забезпечити нову приховану або напівявну функцію.

Приховані функції найбільш ефективні в руках спеціалістів галузі. Вони є дуже цінним інструментом для розслідування, але фальшивомонетник зможе скопіювати багато простіших функцій, якщо їх не вміло застосувати і їхні деталі зберігати в секреті. Однак, завдяки фантазії та винахідливості технолога та дизайнера, можливості є майже необмеженими, а витрати можуть бути мінімізовані або навіть зведені нанівець при внутрішньому застосуванні. Внутрішнє застосування також має переваги, оскільки обмежує участь сторонніх постачальників, які можуть не заслуговувати на довіру в деяких середовищах. Лише найбезпечніші приховані функції можна безпечно використовувати в відкритому контексті, і вони зазвичай підпадають під наступний заголовок криміналістичних маркерів.

КРИМІНАЛІСТИЧНІ маркери

Існує широкий спектр високотехнологічних рішень, які потребують лабораторних випробувань або спеціальних тестових наборів для наукового підтвердження автентичності. Це суто піднабір прихованих технологій, але різниця полягає в науковій методології, необхідній для автентифікації.

Приклади:

Хімічні мітки

Сліди хімічних речовин, які можуть бути виявлені лише високоспецифічними системами реагентів, але зазвичай не виявляються звичайним аналізом.

Біологічні мітки

Біологічний маркер може бути включений у дуже низькій кількості (частки на мільйон або менше) у складі продукту чи покриття або непомітно нанесений на компоненти упаковки. На таких низьких рівнях їх неможливо виявити за допомогою звичайних аналітичних методів, і для їх автентифікації потрібні спеціальні набори реагентів «замок і ключ».

ДНК-мітки

Високоспецифічні ДНК-системи реагентів «замок і ключ» можуть бути застосовані до упаковки різними методами друку. Їм потрібен рекомбінантний ланцюг «дзеркального відображення», щоб здійснити сполучення, і цю реакцію можна виявити спеціальним пристроєм. Безпека додатково забезпечується приховуванням пари маркера та реагенту в матриці випадкових ланцюгів ДНК, але тест налаштований на роботу лише з однією рекомбінантною парою.

Ізотопні співвідношення

Ізотопи, що зустрічаються в природі, можуть бути дуже характерними для джерела сполуки та точно визначені методами лазерної флуоресценції або магнітного резонансу. Вони можуть забезпечувати «відбиток» одного чи кількох компонентів продукту, або альтернативно можна додати певний маркер із власним унікальним підписом. Для виявлення потрібне високоспеціалізоване лабораторне обладнання

Мікротеги

Мікромітки — це мікроскопічні частинки, що містять закодовану інформацію для однозначної ідентифікації кожного варіанту шляхом дослідження під мікроскопом. Це може мати форму буквено-цифрових даних, зображених на невеликих пластівцях або нитках, або фрагментів різнокольорових багатошарових ламінатів із характерною комбінацією кольорів. Їх можна вставляти в клеї або безпосередньо наносити на компоненти упаковки у вигляді плям або ниток.

Є кілька дуже надійних і безпечних варіантів, які можуть зробити їх використання більш відомим і, отже, доступним для довірених органів і слідчих. Однак вони, як правило, підпадають під патентний захист і тому обмежені щодо доступності та ціни.

Технології серіалізації/TRACK і TRACE

Низка додатків Track and Trace розробляється для фармацевтичного сектору, хоча принципи були встановлені протягом багатьох років в інших контекстах. Вони передбачають присвоєння унікальної ідентифікації кожній одиниці запасу під час виробництва, яка потім залишається з нею через ланцюжок постачання до її споживання. Ця ідентичність, як правило, включатиме відомості про назву продукту та силу, а також номер партії та термін придатності, хоча в принципі це може бути просто у формі унікального коду упаковки, який забезпечує доступ до тієї самої інформації, що зберігається в безпечній базі даних.

(Це останнє рішення долає деякі занепокоєння щодо конфіденційності, коли закодовані дані можуть бути прочитані на відстані за допомогою радіообладнання.)

Вони виконують кілька різних функцій:

(a) Відстеження товару через ланцюжок постачання до кожної точки, де є засоби для збору даних.

(b) Забезпечення відстеження історії будь-якого предмета (електронний родовід) з обмеженням кількості контрольних точок.

(c) Увімкніть автентифікацію даних у будь-який час, і непрямо, пакета або блоку, до якого вони застосовані.

Найбільш очевидні переваги – це логістика постачання, де більша прозорість запасів і моделей попиту може призвести до підвищення ефективності та зниження витрат. Іншою перевагою є можливість ідентифікації продукту аж до його видачі пацієнту, що дозволяє усунути помилки при лікуванні та здатність швидко відкликати дефектні партії продукту. Але можливість жорстко контролювати та перевіряти автентичність усіх продуктів через ланцюжок постачання значно зменшує ймовірність того, що контрафактний, викрадений або перенаправлений продукт потрапить до системи розподілу без виявлення.

Слід також зазначити, що теги або ярлики Track and Trace не обов'язково можуть застосовуватися на рівні одиничної упаковки, а можуть бути обмежені цілими ящиками або навіть піддонами, що забезпечує логістичні переваги, але не всі переваги безпеки та безпеки. Як уже згадувалося раніше, ключовим елементом безпеки є серіалізація упаковки.

Серіалізація

Сама по собі мітка Track and Trace може бути не захищена від копіювання чи фальсифікації, але її безпека значно покращується завдяки включенню унікальної та, очевидно, випадкової серіалізації або непослідовної нумерації, в ідеалі на рівні окремих елементів. Якби серіалізація була послідовною, то рівень безпеки був би дуже низьким, оскільки послідовність передбачувана, тоді як «випадкова» серіалізація з використанням високонадійного алгоритму або методу шифрування долає це. Окремі пакети все ще можуть бути скопійовані, але база даних ідентифікуватиме дублікати або недійсні серійні номери, а також ті, які були скасовані чи прострочені, або які з'являються на неправильному ринку, або з недійсними даними про продукт.

Якщо захищена серіалізація застосована до пакета видимо, клієнти можуть перевірити його автентичність за допомогою телефонного або інтернет-посилання на базу даних. Одна з проблем, яку потрібно вирішити, — це право власності, управління та доступ до бази даних, щоб гарантувати, що інформація є легкодоступною та водночас захищеною від компрометації.

Існує два основних механізми включення унікальних пакетних даних для полегшення автоматичного збору даних:

Штрих-коди

Це лінійні або двовимірні штрих-коди високої щільності, що містять ідентичність продукту аж до рівня упаковки, які скануються та посилаються на центральну базу даних.

Однією з популярних реалізацій є код 2D datamatrix, а також інші можливості

Коди PDF417. Двовимірний код, як правило, може мати квадратний розмір 1 см або менше, але все ж містити до 1 Кб даних із певною «надлишковістю» або виправленням помилок. Якщо простір не є обмеженням, також можна використовувати лінійні штрих-коди. Коди можна роздрукувати за допомогою методів онлайн, включаючи струменевий або цифровий друк, що дозволяє здійснювати пряме керування комп'ютером і передавати записи до центральної бази даних. Розробляються ієрархічні системи, за допомогою яких етикетка на ящику для транспортування нерозривно пов'язана з ідентичністю всього його вмісту, і це може далі поширюватися вгору по ланцюжку до етикеток на піддонах, таким чином подолавши необхідність лінії сканування місця через ланцюг постачання.

Так звані технології «нанодруку» дозволяють мікроскопічне нанесення на окремі планшети. УФ-чорнила дозволяють невидимий друк на будь-якій основі, включаючи скляні флакони та ампули.

Тегування радіочастотної ідентифікації (RFID).

RFID-мітка складається з антени з мікрочіпом у центрі. Він містить інформацію про конкретну позицію та партію, яку можна запитати на відстані та без прямої видимості (на відміну від штрих-кодів). Використана радіочастота визначає діапазон і чутливість, але жодна специфікація не підходить для всіх застосувань. Деякі системи можуть фіксувати кілька записів для суміші різних продуктів, але існують деякі проблеми щодо орієнтації міток і поглинання радіосигналу рідинами та фольгою. Але однією очевидною перевагою RFID є те, що він має потенціал для повної автоматизації на складах і навіть в аптеках, не вимагаючи ручного втручання.

Розробляються специфікації обладнання та стандарти даних. Вартість міток залишається суттєвою перешкодою для застосування окремої упаковки, так само як і доступність програмного забезпечення та обладнання для перевірки, якщо воно буде впроваджено на аптечному рівні.

Іншою проблемою є стійкість міток під час нанесення та обробки до кінця терміну служби, оскільки проведені на сьогоднішній день випробування вказують на значний відсоток відмов. Проте є оптимізм, що друкована версія може бути розроблена. Питання конфіденційності та сприйнятливість до навмисної фальсифікації також повинні бути розглянуті перед широким впровадженням.

Унікальне маркування поверхні або рельєф.

Існує кілька методів застосування псевдовипадкового зображення до кожного елемента в партії, наприклад візерунка з ліній або крапок в одній частині коробки, а потім сканування підпису в базу даних пакета за допомогою безпечних алгоритмів для подальшої автентифікації.

Крім того, поверхня упаковки забезпечує унікальний відбиток пальця під час сканування спеціальним лазерним пристроєм, що дозволяє реєструвати кожну упаковку в базі даних під час серійного виробництва, і яку неможливо відтворити або підробити.

Унікальна серіалізація упаковок має потенціал для забезпечення надійних рішень проти шахрайства та підробки фармацевтичних препаратів, але ще не повністю розроблена. Системи штрих-кодів використовують перевірену існуючу технологію, але не мають переваг автоматизації та дистанційного сканування, можливих за допомогою RFID. Але системи RFID ще не перевірені чи надійні, і стандарти потрібно узгодити та визначити. Мітки RFID можуть бути вразливими до навмисної та невидимої зміни або пошкодження.

Як можна побачити вище, існує величезний діапазон можливих рішень, починаючи від дуже простих до дуже складних, від нульових витрат до дуже

дорогих і від крихких до дуже надійних проти компромісу. Широкий вибір опцій підвищує потенційну безпеку, зменшуючи переваги, отримані фальшивомонетником у знищенні будь-якої однієї системи, і виробникам слід вибирати широкий і мудрий вибір для оптимального підвищення безпеки.

Малоймовірно, що якесь одне рішення підійде для всіх застосувань - вартість може бути недосяжною на ринках, що розвиваються, або для продуктів з низькою націнкою, включаючи генерики та безрецептурні препарати.

Фармацевтичне виробництво не обмежується високорозвиненими та складними суспільствами, а є майже універсальним. Таким чином, не всі райони мають однаковий доступ до технологічних рішень та інфраструктури їх постачання. Також зазначається, що в деяких регіонах із слабкою історією захисту інтелектуальної власності може бракувати надійних і безпечних джерел постачання. На таких територіях виробники можуть бути обмежені використанням лише власних технологій.

Практично всі доступні рішення пов’язані з деякими витратами та адміністративним тягарем, тоді як комерційне обґрунтування виробників щодо співвідношення витрат і переваг надзвичайно важко кількісно оцінити. Цьому не допомагають численні необґрунтовані заяви про рівень підробок на світовому ринку ліків. Справжнє ділове обґрунтування більш реалістично базується на управлінні ризиками та корпоративній етичній відповідальності за громадське здоров’я та безпеку, за винятком тих небагатьох сфер, де рівень підробок можна виміряти.

Нарешті, не існує єдиного рішення для кожної проблеми, і безпечна стратегія майже напевно включатиме суміш технологій, часто в комбінації. Відкрита функція майже напевно включатиме захищений прихований елемент для додаткової безпеки, і будь-який продукт може мати кілька різних функцій на різних рівнях упаковки та компонентів. Але до тих пір, поки фальшивомонетники

націлені на ліки для незаконної вигоди, продукт без будь-якої форми антипідробного маркера становить значний потенційний ризик для здоров'я та безпеки населення.

Рішення, які можна перевірити користувачем, були б ідеальним варіантом, якби вони були універсально надійними, доступними та зрозумілими кінцевим користувачам. Деякі ліцензовані технології стверджують, що досягають цього, але обов'язкове їх використання було б контрпродуктивним. Вони можуть не підходити для всіх застосувань і не бути доступними для всіх виробників для всіх продуктів, і їх широке використання стане більшим стимулом для фальшивомонетників інвестувати в розробку технології, як це сталося з голограмами.

Явні функції слід використовувати лише на розсуд виробників. Слід заохочувати їх використання там, де відомо, що продукти та/або ринки знаходяться під загрозою, а в разі використання виробники повинні інформувати громадськість (включно з оптовиками, дистриб'юторами та медичними працівниками) про засоби, за допомогою яких їх можна автентифікувати. Немає сенсу в обов'язковому використанні відкритого рішення, оскільки фальшивомонетники будуть зобов'язані намагатися подолати або обійти його.

Приховані рішення можуть багато чого запропонувати виробникам, але мало користі для органів влади та широкої громадськості через ризик компромісу, якщо вони широко відомі чи широко використовуються. Однак вони можуть бути дуже економічно вигідними та відносно простими в управлінні.

Слід заохочувати виробників застосовувати приховані маркери для всього асортименту продукції та ринків, щоб забезпечити базові засоби моніторингу ситуації. Однак слід уникати широкого використання однієї або двох простих функцій, оскільки ризик компромісу зростає для всіх, і не слід покладатися лише на них для вирішення поточної проблеми підробок. Виробникам слід поділитися

знаннями про деякі приховані маркери з надійними партнерами в ланцюзі поставок.

Криміналістичниймаркери мають деякі переваги перед простішими прихованими функціями, але, як правило, мають високу вартість, як з точки зору ліцензійних зборів або роялті, так і з точки зору необхідного обладнання. Їх захист може бути достатньо надійним, щоб дозволити відкриту рекламу їх присутності, і вони можуть подолати розрив між менш безпечними прихованими функціями та ненадійними явними функціями.

Слід заохочувати використання криміналістичних маркерів у зонах високого ризику, і як тільки виробник прийме на себе зобов'язання щодо системи, перевага може бути в більш широкому використанні в його портфелі за невеликі додаткові витрати. Однак вибір системи має бути на розсуд виробника, і будь-які спроби обов'язкового рішення повинні бути унеможливлені.

Серіалізація/відстеження та трасуванняСистеми відрізняються тим, що вони не обов'язково захищені від копіювання, але захищають ланцюг поставок від проникнення та зловживань. Вони також мають додаткові переваги щодо безпеки, а вдосконалення логістики постачання та ефективності свідчать про те, що вони можуть самофінансуватися ще до того, як врахують питання безпеки та усунення підробок. Є вагомі підстави вважати, що загальна структура бази даних підтримуватиме будь-яку із запропонованих реалізацій, будь то 2D штрих-код або RFID. RFID багатообіцяюча, але попереду довгий шлях, перш ніж вона стане перевіреною, надійною, доступною та практичною на всіх ринках. Слід також визнати, що проблема підробок є найбільшою на ринках, де найбільше бракує IT-інфраструктури, необхідної для підтримки Track and Trace, а торговці підробками не мають стимулів заохочувати її розвиток.

Рекомендація: виробників і постачальників медичних послуг слід заохочувати до платформ, стандартів і практичних аспектів впровадження

технології Track and Trace. Створення специфікацій для структури даних та інфраструктури бази даних є важливими платформами для гармонізації універсальної системи. Необхідно узгодити принципи власності, управління та доступу, а режими доступу зробити максимально гнучкими. Вибір апаратної платформи слід залишити за виробниками, але для швидкості та економії рекомендується розробити систему на основі штрих-кодів як пріоритет, що дозволить природний перехід до RFID, якщо, коли та де це можливо. RFID-мітки можуть бути ефективнішими на рівні палет і ящиків, але двовимірні штрих-коди доступніші на рівні окремої упаковки. Для визначення стандартів має бути створена загальногалузева робоча група з представництвом фірмових виробників і генериків, оптовиків, дистриб'юторів, фармацевтів і медичних працівників. Слід розглянути, як можна отримати доступ до таких органів, як митниця, поліція та слідчі з охорони здоров'я, а також, зрештою, до клієнта.

1.2 Протоколи транспортного рівня TCP та UDP

Протягом багатьох років користувачі, які мали справу зі стеком TCP/IP, працювали з одним із двох транспортних протоколів: TCP та UDP. Однак настав час, коли для деяких додатків була потрібна функціональність, що виходить за рамки тієї, яку може надати ці два протоколи. У 2000 році організація IETF (Internet Engineering Task Force) схвалила як стандарт протокол передачі з управлінням потоком SCTP (Stream Control Transmission Protocol), який почав активно впроваджуватися в мережах не так давно.

Протокол SCTP був створений у рамках проекту, започаткованого робочою групою SIGTRAN, і в першу чергу призначався для транспортування сигнальної інформації ОКС-7 IP-мережами. Суворі вимоги ОКС-7 до параметрів втрат та дотримання черговості проходження повідомлень призвели до створення нового

транспортного протоколу, вільного від недоліків UDP та TCP. IETF пропонує використовувати його як протокол транспортного рівня загального призначення, що поєднує функції TCP та UDP над рівнем IP, оскільки й інші додатки можуть використовувати деякі можливості цього протоколу.

Як було зазначено, всі три протоколи працюють на транспортному рівні еталонної моделі взаємодії відкритих систем ВОС (OSI), основне завдання якого – реалізація наскрізного зв'язку між вузлами мережі, а також за потреби управління потоком та запобігання перевантаженням мережі. Транспортний рівень компенсує ненадійність, властиву нижнім рівням, за рахунок обробки помилок, які спричинені спотворенням даних, втратою пакетів та їхньою доставкою не по порядку. Протоколи UDP, TCP та SCTP входять до стек TCP/IP, де також працюють на транспортному рівні. Відповідність стека моделі ВОС показано на рис.1.

Протокол UDP

Протокол дейтаграм користувача UDP (User Datagram Protocol) був описаний у документі RFC 768 і прийнятий IETF у 1980. Він не орієнтований на створення з'єднання, його головна відмінність – відсутність гарантії доставки та підтримки впорядкованості повідомлень, що передаються, до місця призначення (порядок повідомлень може бути змінений через особливості роботи та примхи IP-мережі). Ці відмінності – наслідок логіки роботи протоколу Додаток, що використовується UDP, формує один пакет, який передається в IP-дейтаграмі. Якщо дейтаграма дублюється в мережі, то на приймаючий вузол можуть бути доставлені два її екземпляри. Якщо ж клієнт UDP відправляє дві дейтаграми в те саме місце призначення, то їх порядок може бути змінений мережею, і вони будуть доставлені з порушенням вихідного порядку. Тому в додатках, які використовують UDP, розробники повинні реалізовувати функції, які певною мірою компенсують ненадійність цього протоколу: тайм-аути, повторну передачу,

обробку втрачених дейтаграм та порядкові номери для зіставлення відповідей запитам. Цей підхід дозволяє протоколу набагато швидше та ефективніше доставляти дані для додатків, яким потрібна велика пропускна спроможність мережі зв'язку або малий час доставки даних. Але через відсутність контролю перевантажень, зростаючого обсягу неконтрольованого високошвидкісного навантаження та використання загального мережевого інфраструктури з іншими сервісами створюється реальна небезпека навантаження мережі, що веде до значного падіння її продуктивності. Частково цю проблему покликаний вирішити щодо молодого протоколу DCCP (Datagram Congestion Control Protocol), описаного в RFC 4340. Цей протокол – альтернатива UDP для додатків, яким необхідний сервіс одноадресної негарантованої доставки дейтаграм, висока швидкість роботи та реалізовані на транспортному. механізми для відстеження перевантажень у мережі без необхідності створювати їх на рівні додатків. На жаль, обсяг статті не дозволяє зупинятися на ньому докладніше.

Протокол TCP

Протокол управління передачею TCP (Transmission Control Protocol), розроблений на замовлення агентства перспективних дослідницьких програм ARPA, був опублікований у документі RFC 793 у вересні 1981 року, а вже в 1983 повністю замінив собою протокол NCP (Network Control Protocol) у мережі ARPANET (пра- батьку Інтернету) і сьогодні став основним протоколом для передачі даних.

Протокол TCP перед початком передачі даних в обов'язковому порядку встановлює з'єднання і забезпечує надійний упорядкований двосторонній байтовий потік, що використовує його додатків. Він підтримує надсилання та прийом підтверджень, обробку тайм-аутів, повторну передачу, керування потоком та інші можливості, які описані в ряді документів (RFC 1323, 2581, 2988, 3390 та 5681).

Протокол TCP пропонує програмам сервіс надійної та впорядкованої передачі. Всі надіслані дані підлягають обов'язковому підтвердженню зустрічною стороною, причому формуються підтвердження не для кожного конкретного успішно отриманого пакета, а для всіх даних від початку посилки до деякого порядкового номера. Якщо підтвердження не надходить протягом часу RTO (Retransmission Time Out), протокол TCP автоматично передає дані повторно і перезапускає таймер знову. Величина таймера RTO динамічно змінюється і залежить від часу двосторонньої затримки, яка визначається за допомогою спеціальних алгоритмів, типу мережі та конкретної реалізації протоколу. Сумарний час повторних спроб надсилання даних у середньому може тривати до 10 хвилин. Зрозуміло, TCP не може гарантувати отримання даних адресатом, оскільки це неможливо. Якщо здійснити доставку неможливо, TCP повідомляє про це користувача, припиняє спроби повторної передачі та розриває з'єднання. TCP можна умовно вважати протоколом, надійним на всі 100%: він забезпечує доставку даних або повідомлення про невдачу.

Упорядкування даних здійснюється прив'язкою деякого порядкового номера до кожного байта, що відправляється. Припустимо, що програма записує 2048 байт у сокет TCP, що призводить до відправки двох сегментів: перший містить дані з порядковими номерами 1-1024, другий - з номерами 1025-2048. Якщо якийсь сегмент приходить поза чергою, приймаючий TCP перед відправкою даних додатку заново впорядкує сегменти, використовуючи їх порядкові номери. Якщо TCP отримує дубльовані дані, він може їх визначити і відкинути.

Для забезпечення процедури керування потоком (Flow Control) TCP завжди повідомляє зустрічному вузлу, який буферний простір він виділив для прийому даних, і вузол, що відправляє, не може перевищувати цього обмеження. Ця процедура отримала назву "метод ковзного вікна". У будь-який час вікно відповідає вільному простору в буфері одержувача. Даний метод гарантує, що

відправник не переповнить цей буфер, а також дозволяє оптимізувати та прискорити процес передачі великих обсягів даних. Вікно змінюється динамічно в міру зчитування приймаючим додатком даних із буфера, а значення розміру передається відправнику разом із повідомленням про підтвердження. Якщо приймаючий буфер TCP заповнений, то можлива ситуація, коли розмір вікна стане нульовим. У цьому випадку відправник змушений чекати, поки одержувач не рахує дані з буфера. За потреби дані можна "проштовхнути", використовуючи функцію PUSH. Функція запускається установкою в повідомленні прапора PSH, після чого всі дані з таким прапором і дані в буфері одержувача будуть передані додатку, що приймає.

Додаток, що використовує TCP, має можливість надсилати та приймати дані в обох напрямках на заданому з'єднанні у будь-який момент часу. Інакше кажучи, TCP може працювати в повнодуплексному режимі і повинен відстежувати порядкові номери та розміри вікна для кожного напряму потоку даних. Після встановлення двостороннього з'єднання воно може бути перетворено на одностороннє.

Протокол TCP, що підтримується практично всіма додатками Інтернету, за минулі роки був значно вдосконалений для забезпечення надійності та продуктивності в мережах різної ємності та якості. Тим не менш, у ньому збереглися властивості, які роблять його непридатним для таких завдань, як передача сигнальних повідомлень у VoIP-мережах або асинхронна обробка на базі транзакцій. TCP вимагає наявності служби доставки зі строго впорядкованою передачею для всіх даних, що пересилаються між двома хостами. Це надто серйозне обмеження для додатків, які допускають як послідовну (часткове впорядкування), так і непослідовну доставку повідомлень. TCP трактує кожну передачу даних як неструктуровану послідовність байтів і зберігає ніяких

неявних структур в переданих потоках даних. Додатки, які обробляють окремі повідомлення, повинні додавати в потік байтів межі повідомлень та відстежувати їх.

Заснований на механізмі TCP-сокетів API-інтерфейс не підтримує множинну адресацію, через що програма може пов'язати лише одну IP-адресу іншого вузла з конкретним TCP-з'єднанням. Якщо інтерфейс, призначений цій IP-адресі, відключається, то TCP-з'єднання переривається, і його необхідно встановлювати заново, що вносить суттєві затримки, особливо критичні для додатків реального часу.

Коли потрібно використовувати UDP замість TCP і чому? Ми усвідомлено ставимо це питання до розгляду протоколу SCTP, оскільки завдання порівняння протоколу UDP з протоколом TCP або SCTP нічим не відрізняється через протиставлення ненадійного протоколу протоколам гарантованої доставки.

UDP може використовуватися для програм широкомовної та багатоадресної передачі (надсилання одного пакета кільком одержувачам замість відправлення копій пакета через кілька з'єднань TCP). Якщо потрібна будь-яка форма захисту від втрат і порушення порядку повідомлень, то відповідна функціональність повинна бути додана додатками як на клієнтській, так і на серверній стороні. Однак програми часто використовують широкомовну та багатоадресну передачу, коли деяка кількість втрат цілком припустима (наприклад, втрата аудіо- або відеопакетів). Є додатки, що вимагають надійної доставки, наприклад, пересилання файлів за допомогою багатоадресної передачі, але в кожному конкретному випадку розробник повинен вирішити, чи компенсується виграш у продуктивності, що отримується за рахунок використання багатоадресної передачі, додатковим ускладненням програми для забезпечення надійності з'єднань.

UDP може використовуватися для простих додатків запит-відповідь, тому що не потребує встановлення та розриву з'єднання і тому дозволяє здійснити

обмін запитом та відповіддю лише у двох пакетах. Це можливо за умови, що пакет не перевищує розміру MTU (Maximum Transmission Unit), що використовується в даній мережі на канальному рівні. Але тоді функція виявлення помилок повинна бути вбудована в додаток. Щонайменше це означає включення підтверджень, тайм-аутів та механізму повторних передач. Якщо запити та відповіді мають розумний розмір, то управління потоком буває не суттєвим для забезпечення надійності.

UDP не слід використовувати для передачі великої кількості даних (наприклад, передачі файлів). У цьому випадку керування потоком за допомогою вікна, запобігання переповненню та повільний старт повинні бути вбудовані в додаток разом з переліченими вище функціями. Ці механізми, що здійснюються відправником, служать для визначення поточної пропускної спроможності мережі та дозволяють контролювати ситуацію під час її перевантаження. Досвід, накопичений ще до того, як ці алгоритми були реалізовані наприкінці 80-х років, показує, що протоколи, які не знижують швидкість передачі, посилюють перевантаження мережі.

Протокол SCTP

SCTP створює двосторонню асоціацію між двома кінцевими точками та дає можливість роботи з кількома потоками кожної пари кінцевих точок, а також забезпечує підтримку концепції багатоінтерфейсного вузла на транспортному рівні. Приклад такої архітектури можна побачити на рис.2.

Спочатку SCTP проектувався з урахуванням потреб зростаючого ринку IP-телефонії та призначався, зокрема, для передачі сигнальних повідомлень ОКС-7 через Інтернет. Сервіси, що надаються SCTP, мають багато спільного із сервісами TCP та UDP. Протокол SCTP описується в RFC 4960, а введення в SCTP наводиться в RFC 3286. Незважаючи на принципову різницю між SCTP і TCP, для застосування інтерфейс "точка-точка" майже нічим не відрізняється від

інтерфейсу TCP. Подібно до TCP, протокол SCTP забезпечує додаткам, що взаємодіють по IP-мережі, транспортну службу з гарантією доставки та збереженням порядку проходження пакетів. Протокол успадкував багато функцій, розроблених для TCP за останні три десятиліття, в тому числі можливість контролю перевантаження та відновлення загублених пакетів. Насправді будь-яку програму, що працює за протоколом TCP, можна перевести на SCTP без втрати функціональності. Розглянемо основні властивості протоколу SCTP.

Подібно до TCP, протокол SCTP надає додаткам надійну передачу повідомлень впорядкування даних, управління передачею та двосторонній зв'язок. З'єднання за протоколом SCTP між клієнтом та сервером називається асоціацією (association), оскільки це багатопотоковий протокол, що дозволяє задати кілька IP-адрес і один порт для кожної сторони з'єднання. Термін "асоціація" використовується замість слова "з'єднання" навмисно, тому що з'єднання завжди встановлюється між двома IP-адресами, а асоціація означає взаємодію двох систем, які можуть мати по кілька адрес. Кожна із сторін асоціації в даному контексті називається кінцевою точкою (endpoint). Наявність у неї декількох IP-адрес дозволяє забезпечити додаткову стійкість у разі відмови мережі. Надлишкові IP-адреси кінцевої точки можуть відповідати власному з'єднанню з постачальником послуг Інтернету ISP. У такій конфігурації SCTP дозволить обійти проблему, що виникла на одному з адрес, завдяки переключенню на іншу адресу, заздалегідь пов'язану з цією асоціацією SCTP. Для скорочення затримок, спричинених перемиканням з первинного спрямування на альтернативні, використовується механізм контролю працездатності, який отримав назву "серцебиття" (heartbeat). Поки йде передача даних за первинним напрямом, протокол SCTP надсилає пакети контролю працездатності на адреси, які перебувають у режимі очікування. Протокол декларує, що IP-адреса буде відключена, як тільки вона досягне порогового значення неповернених

підтверджень про працездатність. Подібної стійкості можна досягти і в TCP, якщо скористатися протоколами маршрутизації. Наприклад, BGP-з'єднання всередині домену (iBGP) часто використовують адреси, які призначаються віртуальному інтерфейсу маршрутизатора як сторони з'єднання TCP. Протокол маршрутизації домену гарантує використаннябудь-якого доступного шляху між двома маршрутизаторами, що неможливо, якщо адреси, що використовуються, належать інтерфейсу в мережі, де виникли проблеми. Функція множинної адресації SCTP дозволяє вузлам, а не лише маршрутизаторам, використовувати аналогічний підхід, причому навіть із підключеннями через різних провайдерів, що неможливо при використанні TCP з маршрутизацією. Вибраний підхід робить сервер вразливим, коли клієнт відкриває асоціацію, але жодних даних не передає. Для такого клієнта буде виділено ресурси, які він не використовує, а невдалий збіг обставин може призвести до DoS-атаки з боку неактивних клієнтів. Для запобігання подібним ситуаціям до SCTP було додано функцію автоматичного закриття асоціацій (autoclose). Вона дозволяє кінцевій точці SCTP задавати максимальну тривалість бездіяльності асоціації, яка вважається такою, якщо по ній не передаються жодні дані в жодному напрямку. Якщо тривалість бездіяльності перевищує встановлене обмеження, асоціація автоматично закривається реалізацією протоколу. SCTP надає розмежування окремих записів у переданому потоці повідомлень. На відміну від TCP, протокол SCTP орієнтований не так на потік байтів, але в повідомлення. Він забезпечує впорядковану доставку даних і зберігає межі повідомлень у пакетах програми, розміщуючи повідомлення в одній або декількох структурах даних SCTP, які називаються "фрагментами" (chunk). Декілька повідомлень можуть об'єднуватися в один фрагмент, а довге повідомлення може бути сегментовано відразу за декількома фрагментами.

Володіючиможливістю формування кількох потоків повідомлень між кінцевими точками асоціації, протокол SCTP дозволяє контролювати для кожного з них надійність та порядок повідомлень. Завдяки цій властивості усувається можливе блокування лінії типу head-of-line, властиве протоколу TCP, оскільки втрата повідомлення в одному з потоків не блокує доставку повідомлень по інших. Цей підхід прямо протилежний тому, що є в TCP, де втрата єдиного байта блокує доставку всіх наступних байтів по з'єднанню доти, доки ситуація не буде виправлена.

За допомогою SCTP програми можуть використовувати різні моделі доставки, у тому числі строгий порядок передачі (як TCP), часткове впорядкування (за потоками) та невпорядковану доставку (як UDP). Це було зроблено, оскільки деякі додатки не потребують збереження порядку повідомлень під час передачі їх через мережу. Раніше додатку, що використовує TCP для забезпечення надійності, доводилося миритися із затримками, викликаними блокуванням черги та необхідністю впорядкованої доставки (хоча сама програма її може і не потребувати). На це варто звернути особливу увагу, оскільки для SCTP існує різниця між надійною та впорядкованою доставкою. При роботі з TCP ці дві властивості нерозривно пов'язані, оскільки всі дані надійно доставляються (наприклад, загублені пакети передаються повторно) вузлу-одержувачу та надаються додатком у тій послідовності, якою вони передавались. Навпаки, в SCTP ці властивості між собою пов'язані. Номер послідовності в заголовку SCTP гарантує, що всі повідомлення надійно доставляються вузлу-одержувачу, але SCTP передбачає низку варіантів того, в якому порядку представляти повідомлення додатку-одержувачу. Це може бути номер потоку, який застосовується для впорядкування повідомлень по потоках, або передача даних додатку в міру їх появи на вузлі-одержувачі. І знову-таки цей підхід дозволяє усунути затримку, спричинену блокуванням внаслідок неправильного порядку

доставки пакетів. Для відновлення втрачених пакетів використовується схема вибіркового підтвердження, успадкована з TCP. Підтримуючи зворотний зв'язок з відправником, приймач SCTP повідомляє, які пакети необхідно надсилати повторно, якщо вони були втрачені. Завдяки сервісу часткової надійності, відправник отримує можливість вказувати час життя кожного повідомлення. Якщо часткова надійність підтримується обома вузлами, то недо- влені вчасно дані можуть скидатися транспортним рівнем, а не додатком, навіть якщо вони були передані та втрачені. Таким чином, оптимізується передача даних в умовах завантажених ліній.

Для контролю перевантаження використовуються стандартні методики, що вперше застосовувалися ще в TCP, у тому числі повільний старт, запобігання перевантаженню та швидка повторна передача. Додатки SCTP можуть отримати свою частку мережевих ресурсів, співіснуючи з додатками TCP в одній мережі.

Хоча протокол SCTP і розроблявся передачі сигнальних повідомлень ОКС-7 через IP-мережа, у процесі розробки область його застосування значно розширилася. Фактично він перетворився на загальноцільовий транспортний протокол. Будучи еволюційним розвитком протоколу TCP, SCTP підтримує майже всі його функції та значно розширює їх новими сервісами транспортного рівня. Ці сервіси мають важливе значення для передачі повідомлень телефонної сигналізації через мережу IP і водночас можуть забезпечувати низку переваг для інших додатків, які потребують надійного транспортного механізму з високим рівнем продуктивності.

SCTP позбавлений двох особливостей TCP. Одна з них – стан неповного закриття з'єднання на своїй стороні. Цей стан виникає, коли програма закриває свій кінець з'єднання, але дозволяє зустрічній стороні надсилати дані та приймає їх. Додаток входить у цей стан для того, щоб повідомити співрозмовника, що відправка даних завершено. Програми дуже рідко використовують цю

можливість, тому при розробці SCTP вирішено не дбати про її підтримку. Також не підтримується обробка позачергових даних (urgent data). Для доставки термінових даних до SCTP можна використовувати окремий потік, хоча це й не дозволяє точно відтворити поведінку TCP у даній ситуації.

Протокол SCTP забезпечує явну підтримку багатоінтерфейсних вузлів (див. мал.2), що дозволяє значно підвищити рівень надійності асоціації та зменшити затримки, що виникають у разі відмови у доступі або збоїв у магістральній мережі. Але при цьому чинна редакція протоколу SCTP не підтримує розподіл навантаження (load sharing) на альтернативні напрямки, тому дана можливість лише забезпечує надмірність напрямків передачі для підвищення рівня надійності. У IETF зараз ведуться роботи в даному напрямку, і документ, що має версію 7 та статус експериментального, було прийнято у жовтні 2013 року.

Нарешті, TCP-вузли сприйнятливі до атак типу "відмова в обслуговуванні" DoS (Denial of Service), що викликано недосконалістю механізму встановлення з'єднання в TCP, який отримав назву триразового рукостискання (three-way handshake). Для таких атак характерні свого роду "шторми", величезна кількість пакетів TCP SYN, що сигналізують хосту, що нічого не підозрює, про те, що відправник хоче встановити з ним TCP-з'єднання. Хост-одержувач резервує пам'ять та відповідає на запит повідомленнями SYN ACK. Коли атакуюча система не повертає повідомлення ACK, необхідні для завершення триразової процедури встановлення TCP-з'єднання, ресурси хоста, що зазнав атаки, залишаються не звільненими. Тому він не готовий до обслуговування легітимних запитів на встановлення TCP-з'єднання. При розробці SCTP цей недолік був усунений у механізмі чотирикратного рукостискання – сервер резервує необхідні ресурси лише після проходження процедури cookie.

Порівняння можливостей протоколівРезультати аналізу, проведеного в цій статті, наведено в таблиці.

На закінчення можна зробити низку висновків.

Протокол UDP не забезпечує надійності доставки та не має жодних механізмів підтвердження передачі. Однак існують програми, в яких UDP використовувати ціле- гідніше, ніж TCP або SCTP. Маючи високу швидкість передачі та простоту реалізації, UDP ідеально підходить для багатоадресного трафіку, сервісів потокової передачі аудіо- та відеоінформації, розрахованих на багато користувачів ігор у реальному часі та деяких протоколів сигналізації, що використовуються в мережах VoIP. При реалізації механізмів підвищення надійності на рівні програми це завдання може вирішуватися на канальному рівні, компенсуючи недоліки протоколу за рахунок створення надлишкової смуги пропускання. Вже зараз на рівні доступу мідними та оптичними лініями зв'язку можна використовувати канали зі швидкістю до 10 Гбіт/с, що здавалося фантастикою за часів розробки перших протоколів транспортного рівня.

На відміну від UDP, TCP – складніший у реалізації потоковий протокол, який має ряд недоліків, які ми розглянули вище. Як і TCP, SCTP забезпечує надійність передачі, але окрім цього він дозволяє задавати межі повідомлень, забезпечує підтримку множинної адресації на транспортному рівні та пропонує розширені можливості цього рівня, що виходять за рамки тих, які можуть зараз надати TCP та UDP. Багато функцій TCP підтримуються і в SCTP: повідомлення про прийом, повторна передача втрачених даних, збереження послідовності даних, віконне управління передачею, повільний старт та алгоритми запобігання навантаженню, а також вибіркові повідомлення. Протокол SCTP дозволяє застосуванню налаштовувати транспортний рівень за своїми потребами, причому налаштування виконується для кожної асоціації окремо. Ця гнучкість у поєднанні

з універсальним набором значень за промовчанням (для додатків, які не потребують тонкого налаштування транспортного рівня) дає додатку переваги, які вона не могла отримати при роботі з TCP. Протокол SCTP активно впроваджується на мережах зв'язку в ширшій області, ніж та, для якої він створювався, але через пасивність розробників додатків та операційних систем, не тими темпами, на які розраховували його творці

2. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ

2.1 Аналіз мови програмування C++

C++ — це широко використовувана мова програмування загального призначення, за допомогою якої можна писати високоефективні програми.

Через це він також популярний у критично важливих для безпеки сферах застосування, наприклад в автомобільній промисловості, де MISRA є одним із найпопулярніших стандартів кодування.

C++ був винайдений датським комп'ютерним науковцем Б'ярне Страуструпом з AT&T Bell Labs у 1979 році. Він виник на основі аналізу ядра UNIX, щоб дослідити, якою мірою його можна поширювати в мережі.

Коли Страуструп працював над своєю докторською дисертацією в обчислювальній лабораторії Кембриджського університету, він був вражений програмною організацією та функціями паралелізму мови програмування Simula, яку він використав для написання симулятора. Однак він виявив, що реалізація погано масштабується, тому зрештою симулятор був переписаний на BCPL.

Для своєї роботи в AT&T Bell Labs Страуструп вирішив розширити мову програмування C за допомогою мовних функцій, подібних до тих, які він знайшов такими корисними в Simula. Він почав писати препроцесор Cpre , який перетворював програми C із класами, подібними до Simula, у звичайний код C, який можна було скомпілювати за допомогою існуючих компіляторів. Нова мова спочатку називалася просто «C з класами».

З самого початку мета полягала в тому, щоб нову мову можна було використовувати для всього, для чого можна використовувати C, щоб це була мова програмування загального призначення. Крім того, оскільки компілятори C вже

були доступні для багатьох платформ, він успадкував портативність C, яка донині є одним із важливих атрибутів якості. Іншою метою мови було надання кращих альтернатив для небезпечних функцій C, зберігаючи при цьому його ефективність і прямий доступ до основних апаратних функцій.

C з наданими класами:

- Заняття
- Похідні класи
- Публічний/приватний контроль доступу
- Конструктори та деструктори
- Функції виклику та повернення (незабаром видалено через недостатню популярність)
- Дружні заняття
- Перевірка типу аргументів функції
- Вбудовані функції
- Аргументи за замовчуванням
- Перевантаження оператора присвоювання.

C++

На цьому етапі історії C++ мова потребувала належної назви. Деякий час він називався C84, але це вважалося потворним і заплутаним. Зрештою комп'ютерний вчений Рік Маскітті запропонував назву C++, яку можна інтерпретувати як мову, яка є наступницею C.

Оскільки до мови було додано більше можливостей, попередній процесор Cpre більше не був придатним, і була написана правильна назва компілятора Cfront . Він все ще виробляв код C для зручності, але це був належний компілятор, оскільки він виконував повну перевірку синтаксису та семантики, а також

створював внутрішнє представлення програми з однією таблицею символів на область.

Нові мовні функції включали:

- Віртуальні функції
- Перевантаження імені функції та оператора
- Список літератури
- Конст
- Контроль вільного зберігання пам'яті, який контролюється користувачем
- Покращена перевірка типів і коментарі стилю C++ (які фактично були взяті з BCPL).

У 1986 році була опублікована перша редакція книги "Мова програмування C++", яка описувала мову відповідно до компілятора Cfront 1.0.

C++ випуск 2.0

Друга версія мови була завершена в 1989 році та підвищила стабільність її визначення та реалізації.

Додано C++ 2.0:

- Множинне успадкування
- Типобезпечне з'єднання
- Покращена роздільна здатність перевантажених функцій
- Рекурсивне визначення присвоювання та ініціалізації
- Покращено можливості для керування пам’яттю, що визначається користувачем
- Абстрактні заняття
- Статичні функції-члени
- Функції-члени Const
- Захищені члени

– Перевантаження оператора -> та покажчиків на члени.

C++ випуск 3.0

Це була остання версія C++ до стандартизації мови. C++ 3.0 був завершений у 1991 році та додав шаблони класів і функцій. У 1993 році мав вийти випуск C++ 4.0 із додаванням обробки винятків, початкову реалізацію якої було зроблено компанією Hewlett-Packard у 1992 році, але це так і не було завершено.

Анотований довідковий посібник C++

Плани AT&T щодо нового компілятора C++ так і не здійснилися, тоді як з'явилися інші компілятори C++, як комерційні (включаючи Borland, IBM, DEC і Microsoft), так і компілятор GNU з відкритим кодом g++. У результаті Страуструп зосередився на розробці та стандартизації мови. Annotated C++ Reference Manual , опублікований у 1991 році, став відправною точкою для стандарту мови. Посібник містив повне визначення C++, а не лише функції, реалізовані Cfront 3.0, і був переглянутий багатьма людьми з різних організацій. Новими функціями були простори імен, вкладені класи та обробка винятків.

C++98

Стандартизація ANSI C++ була розпочата в 1989 році компанією Hewlett-Packard спільно з AT&T, DEC і IBM. Стандартизація мови стала необхідною з кількох причин: додавання важливих нових функцій і запобігання розвитку несумісних діалектів. У 1991 році почалася стандартизація ISO, і з того часу комітети проводять спільні засідання.

Важливим заходом було визначення стандартної бібліотеки, включаючи стандартну бібліотеку шаблонів (STL). Крім того, він додав:

- Інформація про тип у реальному часі (RTTI: dynamic_cast, typeid)
- Коваріантні типи повернення
- Оператори приведення
- Змінний
- Bool
- Декларації в умовах
- Шаблони учасників
- Ініціалізатори членів класу
- Окреме складання шаблонів (експорт)
- Шаблон часткової спеціалізації

– Часткове впорядкування шаблонів функцій перевантаження.

C++03 і вбудований C++

C++03 був технічним випуском C++98 зі змінами, затвердженими для технічного виправлення. Комітет також почав думати про C++0x.

Тим часом консорціум японських розробників інструментів для вбудованих систем, включаючи Toshiba, Hitachi, Fujitsu та NEC, запропонував підмножину Embedded C++ (EC++). Це було призначено для програмування вбудованих систем. Підмножина видалила мовні функції, які могли погіршити продуктивність або сприймалися розробниками як надто складні, і тому розглядалися як небезпека для продуктивності чи правильності.

Заборонені функції були множинним успадкуванням, шаблонами, винятками, RTTI, приведеннями нового стилю та просторами імен. Крім того, зі стандартної бібліотеки було видалено STL і локалі, а також надано альтернативу для iostreams. Цікаво, що EC++ використовувався не так часто, а наднабір «Ectended EC++», який додав шаблони, був більш популярним.

У відповідь на EC++ комітет опублікував технічний звіт про продуктивність. Технічний звіт про продуктивність надав модель накладних витрат часу та простору, пов’язаних із використанням різних функцій мови та бібліотеки C++. При цьому було вирішено питання про проблеми продуктивності. Крім того, було представлено методи ефективного впровадження. Отже, комітет ISO не схвалив EC++.

C++11

Ця версія представила багато нових основних функцій, так що для багатьох програмістів вона відчула себе новою мовою!

Додано C++11:

– Модель пам'яті

– Паралелізм

- Auto і decltype
- Діапазон-за
- Переміщення семантики та посилань на rvalue
- Рівномірна ініціалізація
- Nullptr
- Функції Constexpr
- Визначені користувачем літерали
- Необроблені рядкові літерали
- Атрибути
- Лямбда
- Варіативні шаблони
- Псевдоніми шаблонів (з використанням)
- Ні, крім
- Перевизначення та остаточне
- Static_assert
- Довгий довгий
- Ініціалізатори членів за замовчуванням
- Ініціалізація в конструкторі
- Класи Enum.

Були також значні доповнення до стандартної бібліотеки. У 1998 році була заснована організація Boost, яка надає безкоштовні рецензовані портативні вихідні бібліотеки C++. Бібліотека Boost була важливою, тому що різні функції бібліотеки були доступні в ній на ранніх стадіях, щоб стандарт ISO міг скористатися досвідом, отриманим від їх використання. Модель пам’яті була важливою основою для підтримки паралелізму, яка забезпечувала потоки та блокування.

Семантика переміщення може підвищити ефективність, оскільки усуває непотрібні копії, які можуть бути дорогими для великих об’єктів. Він завершує контроль над терміном служби об’єкта та керування ресурсами, дозволяючи розробнику контролювати, чи ресурс копіюється, чи його власність має бути передана іншому об’єкту.

C++14

Комітет ISO C++ мав на меті внести зміни в основні та другорядні випуски, щоб C++14 мав на меті завершити C++11. Він додав:

- Двійкові літерали (0b)
- Розділювачі цифр
- Змінні шаблони
- Виведення типу повернення функції
- Загальні лямбда-вирази
- Локальні змінні у функціях constexpr
- Переміщення захоплення
- Доступ до кортежу за типом
- Визначені користувачем літерали в стандартній бібліотеці.

C++17

Після незначного випуску C++14, C++17 мав бути великим оновленням. На жаль, деякі основні очікувані функції, такі як концепції та співпрограми, не потрапили в цю версію.

Нові основні функції, які зробили це, включають:

- Дедукція аргументу шаблону класу (представляємо посібники з дедукції)
- Структуровані палітурки
- Вбудовані змінні
- Складні вирази

- Явний тест в умовах
- Гарантована відсутність копій
- Більш суворий порядок оцінки вираження
- Auto як тип аргументу шаблону
- Стандартні атрибути для виявлення типових помилок
- Шістнадцяткові літерали з плаваючою комою
- "якщо constexpr".

Деякі з нових функцій є зразковими для збільшення підтримки функціонального стилю програмування. Ключовий елемент для цього вже був наданий лямбда-виразами в C++11, але вирази згортання (зручна нотація для скорочення списку аргументів до одного значення за допомогою оператора) і посібники з дедукції підвищують функціональний смак мови.

C++20

Основні функції, яких не було в C++17, були додані в C++20. Як наслідок, ця версія є більшим кроком вперед, порівнянним з кроком від C++03 до C++11, тож ми можемо сказати, що ця версія є основним оновленням, яким мав бути C++17.

Основні нові функції мови:

Співпрограми

Концепції

Модулі.

Іншими новими можливостями мови є підтримка обчислень під час компіляції, оператор космічного корабля <=>, покращення паралелізму, призначені ініціалізатори та типи класів у параметрах шаблону, що не є типом (також дозволяє використовувати рядкові літерали як параметри шаблону). Крім того, новими стандартними функціями бібліотеки є діапазони, дата, діапазон і формат.

Модулі, нарешті, забезпечують кращий спосіб вираження модульності, ніж на основі попереднього процесора, який включає файловий механізм,

успадкований від C. Співпрограми забезпечують безстековий механізм для асинхронного виконання послідовного коду. Концепції називаються наборами вимог до аргументів шаблону та є частиною інтерфейсу шаблону. Вони дають змогу визначити передбачуване використання шаблонів і значно покращують ясність помилок компіляції, коли обмеження не задовольняються. Це значне покращення порівняно з попередньою практикою використання «Помилка заміни не є помилкою» (SFINAE), яка призводить до довгих і складних помилок компіляції, коли обмеження порушується.

Майбутнє C++

З часів перших кроків у 1979 році C++ пройшов довгий шлях і продовжує розвиватися.

Незабаром буде випущено C++23 з невеликими, але значними коригуваннями, а над C++26 уже почалася робота.

C++ продовжує зростати в популярності, і його використання розширюється, включаючи створення додатків для віртуальної реальності (VR) через Unreal Engine, а також у додатках для криптовалют.

Довіряйте статичному аналізу Perforce для C++

Інструменти статичного аналізу Perforce Helix QAC і Klocwork користуються довірою вже понад 30 років для безпечного, захищеного та високоякісного коду на C, C++ тощо. Наші інструменти виявляють дефекти, уразливості та проблеми відповідності під час кодування та сертифіковані для використання в критично важливих для безпеки програмах.

Helix QAC також надає модулі відповідності для забезпечення дотримання нових інструкцій MISRA C++:2023. Perforce планує отримати повний модуль відповідності MISRA C++:2023 після випуску стандарту.

Дізнайтеся, чому Helix QAC є найкращим статичним аналізатором коду для MISRA C і MISRA C++.

2.2 Аналіз інтегрованих середовищ розробки

Visual Studio - це інтегроване середовище розробки (IDE), розроблене компанією Microsoft для роботи з різними мовами програмування, включаючи C++, C#, Visual Basic і багато інших. Воно включає в себе широкий спектр інструментів для розробки програмного забезпечення для різних платформ, включаючи Windows, веб-додатки, мобільні додатки і багато іншого.

Однією з ключових особливостей Visual Studio є його потужна система редактора коду, яка надає широкі можливості автодоповнення, підказки та перевірки синтаксису під час написання коду. Це робить процес написання коду більш продуктивним і зручним для розробників.

Крім того, Visual Studio має інтегровані інструменти для відлагодження програм, що дозволяють розробникам легко виявляти і виправляти помилки в своєму коді. Відлагодження може бути здійснене на крокування, зупинення виконання програми, а також використання точок зупинки для аналізу стану програми в реальному часі.

Visual Studio також має вбудовану підтримку для різних систем керування версіями, таких як Git і Subversion, що дозволяє розробникам ефективно співпрацювати над проектами та відстежувати зміни в коді.

Крім того, Visual Studio має інтегровані інструменти для роботи з базами даних, тестування програмного забезпечення, створення встановлювачів програм і багато інших. Він надає повний цикл розробки програмного забезпечення, що дозволяє розробникам реалізувати проекти від початку до кінця, усередині одного інтегрованого середовища.

CLion - це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, призначене для роботи з мовами програмування C і C++. Воно створене з урахуванням потреб розробників, що працюють з проектами на цих мовах, і

надає широкий спектр інструментів та функцій для зручної та продуктивної розробки програм.

Однією з ключових особливостей CLion є його потужний редактор коду з високою підтримкою стандартів C і C++. Він має автодоповнення коду, підказки, відстеження використання змінних та інші функції, що допомагають розробникам писати код швидше та ефективніше.

CLion також надає інтегровані інструменти для відлагодження програм, включаючи можливість створювати точки зупинки, крокування коду та аналіз стану програми в реальному часі. Це дозволяє розробникам ефективно виявляти та виправляти помилки в своєму коді.

Крім того, CLion підтримує інтеграцію з різними системами керування версіями, такими як Git, SVN та Mercurial, що дозволяє розробникам легко співпрацювати над проектами та відстежувати зміни в коді.

Важливою особливістю CLion є його підтримка CMake - популярного інструменту для автоматизації процесу збирання програмного забезпечення. CLion надає інтегровані інструменти для налаштування проектів з використанням CMake, що робить процес конфігурації та збирання проектів більш простим та зручним для розробників.

Загалом, CLion - це потужне та ефективне інтегроване середовище розробки для мов програмування C і C++, яке надає розробникам всі необхідні інструменти для успішної розробки програмних проектів.

Code::Blocks - це безкоштовне інтегроване середовище розробки (IDE) для роботи з різними мовами програмування, включаючи C, C++, і Fortran. Воно розроблене для зручної та ефективної розробки програмного забезпечення і має широкий спектр функцій та інструментів для цього.

Однією з ключових особливостей Code::Blocks є його простий та легкий у використанні інтерфейс, що робить його ідеальним вибором як для початківців,

так і для досвідчених розробників. Він має зручну систему керування проектами та можливість редагування та компіляції різних типів файлів.

Code::Blocks має вбудовану підтримку для різних компіляторів, включаючи GCC, MSVC, Clang та інші. Це дозволяє розробникам використовувати їх у своїх проектах та налаштовувати параметри компіляції за необхідності.

Ще однією важливою особливістю Code::Blocks є його розширюваність. Він підтримує плагіни, які дозволяють розширити його функціональність та додати нові можливості, такі як підтримка інших мов програмування, інструменти для відлагодження та інше.

Code::Blocks також має інтегровані інструменти для відлагодження програм, включаючи можливість ставити точки зупинки, крокувати код та аналізувати стан програми в реальному часі. Це дозволяє розробникам ефективно виявляти та виправляти помилки в своєму коді.

Загалом, Code::Blocks - це потужне та ефективне інтегроване середовище розробки для мов програмування C та C++, яке надає розробникам зручні та продуктивні інструменти для розробки програмного забезпечення.

Eclipse CDT (C/C++ Development Tooling) - це розширення інтегрованого середовища розробки Eclipse, яке спеціалізується на розробці програм на мовах програмування C і C++. Це потужне інтегроване середовище розробки, яке надає розробникам широкий набір інструментів для розробки програмного забезпечення для різних платформ.

Однією з ключових особливостей Eclipse CDT є його потужний та гнучкий редактор коду, який має велику кількість корисних функцій, таких як автодоповнення коду, відстеження використання змінних, перевірка синтаксису та багато іншого. Це робить процес написання коду швидшим та ефективнішим для розробників.

Eclipse CDT також має інтегровані інструменти для відлагодження програм, що дозволяють розробникам ефективно виявляти та виправляти помилки в своєму коді. Інструменти відлагодження включають можливість встановлювати точки зупинки, крокувати код та аналізувати стан програми в реальному часі.

Крім того, Eclipse CDT підтримує інтеграцію з різними системами керування версіями, такими як Git, SVN та CVS, що дозволяє розробникам ефективно співпрацювати над проектами та відстежувати зміни в коді.

Особливу увагу слід звернути на підтримку Eclipse CDT роботи з проектами, створеними з використанням інших інструментів розробки, таких як Makefile, CMake та інші. Це дозволяє розробникам працювати з різними типами проектів та використовувати їх у своїй роботі.

Загалом, Eclipse CDT - це потужне та ефективне інтегроване середовище розробки для мов програмування C та C++, яке надає розробникам зручні та продуктивні інструменти для розробки програмного забезпечення.

Таким чином, для розробки програмної реалізації було обрано IDE Visual Studio, що виправдовується його потужними функціональними можливостями, широким спектром інтегрованих інструментів, високою підтримкою стандартів програмування та зручним інтерфейсом користувача. Завдяки цим перевагам, Visual Studio стає ідеальним вибором для розробників, які шукають надійне та ефективне інтегроване середовище розробки для роботи з різними типами проектів на мові програмування C++.

2.3 Діаграма класів

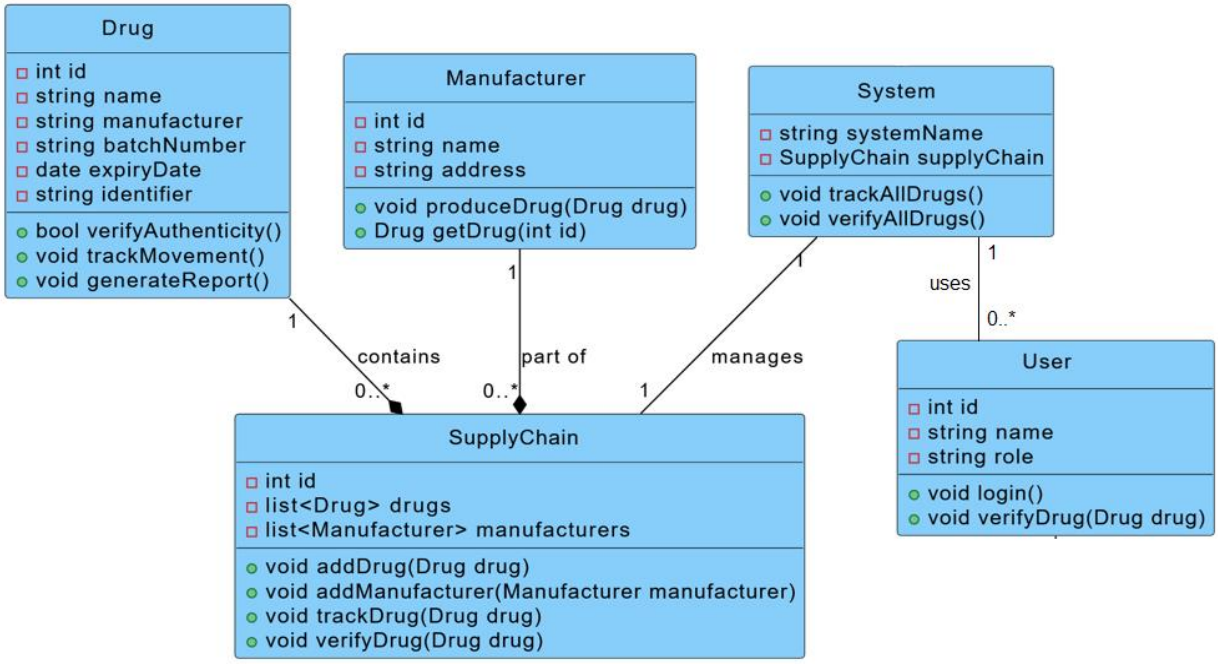


Рис. 2.1 Діаграма класів

Діаграма класів представлено на рисунку 2.1. Ця діаграма представляє структуру системи захисту медикаментів від підробки та нелегального виробництва. Вона включає такі класи: Drug (Ліки), Manufacturer (Виробник), SupplyChain (Ланцюг Постачання), System (Система) та User (Користувач).

Клас Drug містить атрибути, що включають унікальний ідентифікатор (int id), назву ліків (string name), виробника ліків (string manufacturer), номер партії (string batchNumber), дату закінчення терміну придатності (date expiryDate) та унікальний ідентифікатор (string identifier). Методи цього класу дозволяють перевіряти автентичність ліків (bool verifyAuthenticity()), відстежувати їх переміщення (void trackMovement()) та генерувати звіти (void generateReport()).

Клас Manufacturer має атрибути унікального ідентифікатора (int id), назви виробника (string name) та адреси виробника (string address).

Методи класу включають виробництво ліків (void produceDrug(Drug drug)) та отримання ліків за ідентифікатором (Drug getDrug(int id)).

SupplyChain (Ланцюг Постачання) містить атрибути унікального ідентифікатора (int id), списку ліків (list<Drug> drugs) та списку виробників (list<Manufacturer> manufacturers). Методи цього класу дозволяють додавати ліки до ланцюга постачання (void addDrug(Drug drug)), додавати виробників (void addManufacturer(Manufacturer manufacturer)), відстежувати ліки (void trackDrug(Drug drug)) та перевіряти їх автентичність (void verifyDrug(Drug drug)).

Клас System (Система) включає атрибути назви системи (string systemName) та ланцюга постачання (SupplyChain supplyChain). Методи класу дозволяють відстежувати всі ліки у системі (void trackAllDrugs()) та перевіряти їх автентичність (void verifyAllDrugs()).

User (Користувач) містить атрибути унікального ідентифікатора (int id), імені користувача (string name) та ролі користувача в системі (string role). Методи класу включають авторизацію користувача в системі (void login()) та перевірку автентичності ліків (void verifyDrug(Drug drug)).

Ця діаграма ілюструє взаємодію компоненів між собою для забезпечення захисту медикаментів від підробки та нелегального виробництва.

2.4 Діаграма станів

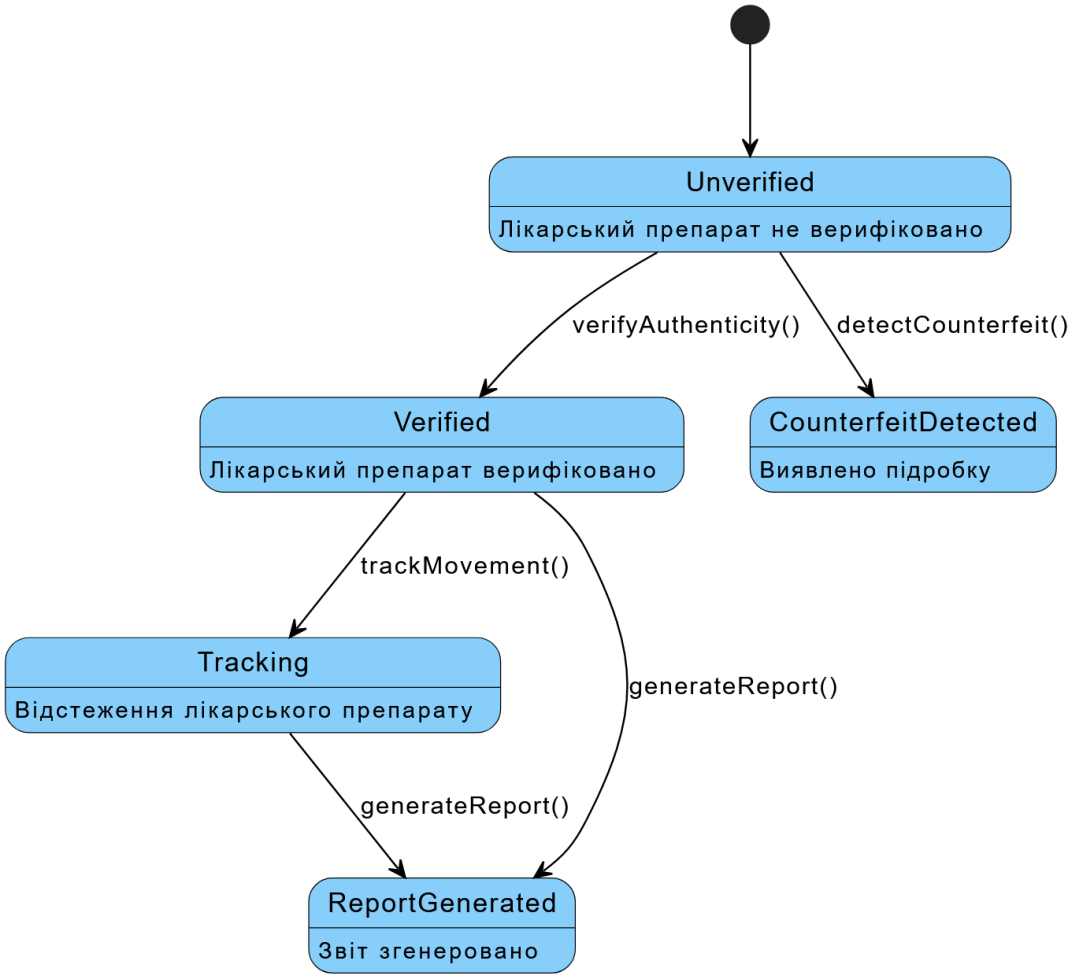


Рис. 2.2 Діаграма станів

Діаграма станів відображають різні стани, через які проходять ліки, та переходи між цими станами на основі певних подій або дій. У початковому стані Unverified (Лікарський препарат не верифіковано) ліки знаходяться до того, як їх автентичність буде перевірена. В цьому стані можливі дві дії: перевірка автентичності (verifyAuthenticity()), яка переводить ліки до стану Verified (Лікарський препарат верифіковано), або виявлення підробки

(detectCounterfeit()), що переводить їх до стану CounterfeitDetected (Виявлено підробку).

Стан Verified означає, що автентичність ліків підтверджена. Після цього ліки можуть бути відстежувані (trackMovement()), що переводить їх до стану Tracking (Відстеження лікарського препарату).

Стан CounterfeitDetected вказує на те, що ліки були визнані підробленими. Подальші дії в цьому стані не передбачені, оскільки подальші кроки можуть включати вилучення або знищення підроблених ліків.

Стан Tracking означає, що ліки відстежуються в системі. У цьому стані можлива дія generateReport(), яка переводить ліки до стану ReportGenerated (Звіт згенеровано).

Стан ReportGenerated означає, що звіт про відстеження ліків був згенерований. Якщо знову виконується дія generateReport(), ліки залишаються у цьому стані.

Таким чином, діаграма ілюструє процес перевірки, відстеження та звітності для ліків у системі, забезпечуючи їх захист від підробки та нелегального виробництва.

2.5 Схема бази даних

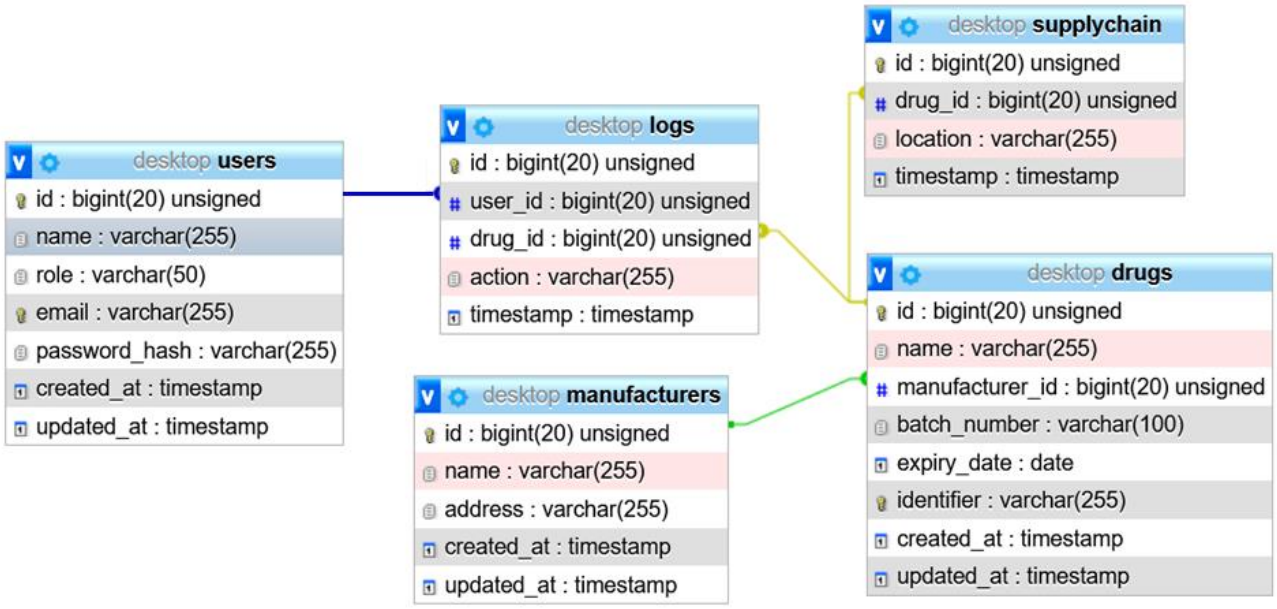


Рис. 2.3 Схема бази даних

Схема бази даних складається з п'яти таблиць: users, logs, manufacturers, drugs та supplychain. Кожна таблиця має свої атрибути та взаємозв'язки з іншими таблицями, що забезпечує повну інтеграцію системи захисту медикаментів.

Таблиця users містить інформацію про користувачів системи, включаючи унікальний ідентифікатор, ім'я, роль, електронну пошту, хеш пароля, а також дату та час створення і останнього оновлення запису. Це дозволяє ідентифікувати кожного користувача та контролювати доступ до системи.

Таблиця logs використовується для запису дій, які виконують користувачі. Вона включає унікальний ідентифікатор запису, ідентифікатор користувача, що виконав дію, ідентифікатор ліків, до яких відноситься дія, опис дії, а також дату та час виконання дії. Ця таблиця пов'язана з таблицею users за допомогою user_id та з таблицею drugs за допомогою drug_id, що дозволяє відстежувати, хто і коли виконував певні дії з ліками.

Таблиця manufacturers містить інформацію про виробників ліків, включаючи унікальний ідентифікатор, назву, адресу виробника, а також дату та час створення і останнього оновлення запису. Це дозволяє зберігати дані про компанії, які виробляють медикаменти.

Таблиця drugs включає деталі про самі ліки, такі як унікальний ідентифікатор, назву, ідентифікатор виробника, номер партії, дату закінчення терміну придатності, унікальний ідентифікатор ліків, а також дату та час створення і останнього оновлення запису. Вона пов'язана з таблицею manufacturers за допомогою manufacturer_id, що дозволяє відстежувати, який виробник виготовив певні ліки.

Таблиця supplychain містить інформацію про переміщення ліків у ланцюгу постачання. Вона включає ідентифікатор ліків, місцезнаходження ліків, а також дату та час запису. Ця таблиця пов'язана з таблицею drugs за допомогою drug_id, що дозволяє відстежувати переміщення кожної партії ліків у ланцюгу постачання.

Ця схема бази даних відображає різні елементи системи які взаємодіють між собою. Користувачі можуть виконувати дії з ліками, які виробляються виробниками, і переміщення ліків відстежується в ланцюзі постачання.

3. РОЗРОБКА ПРОГРАМНОГО РІШЕННЯ

3.1 Розробка програми

При проектуванні програмного рішення було визначено наступну діаграму використання:

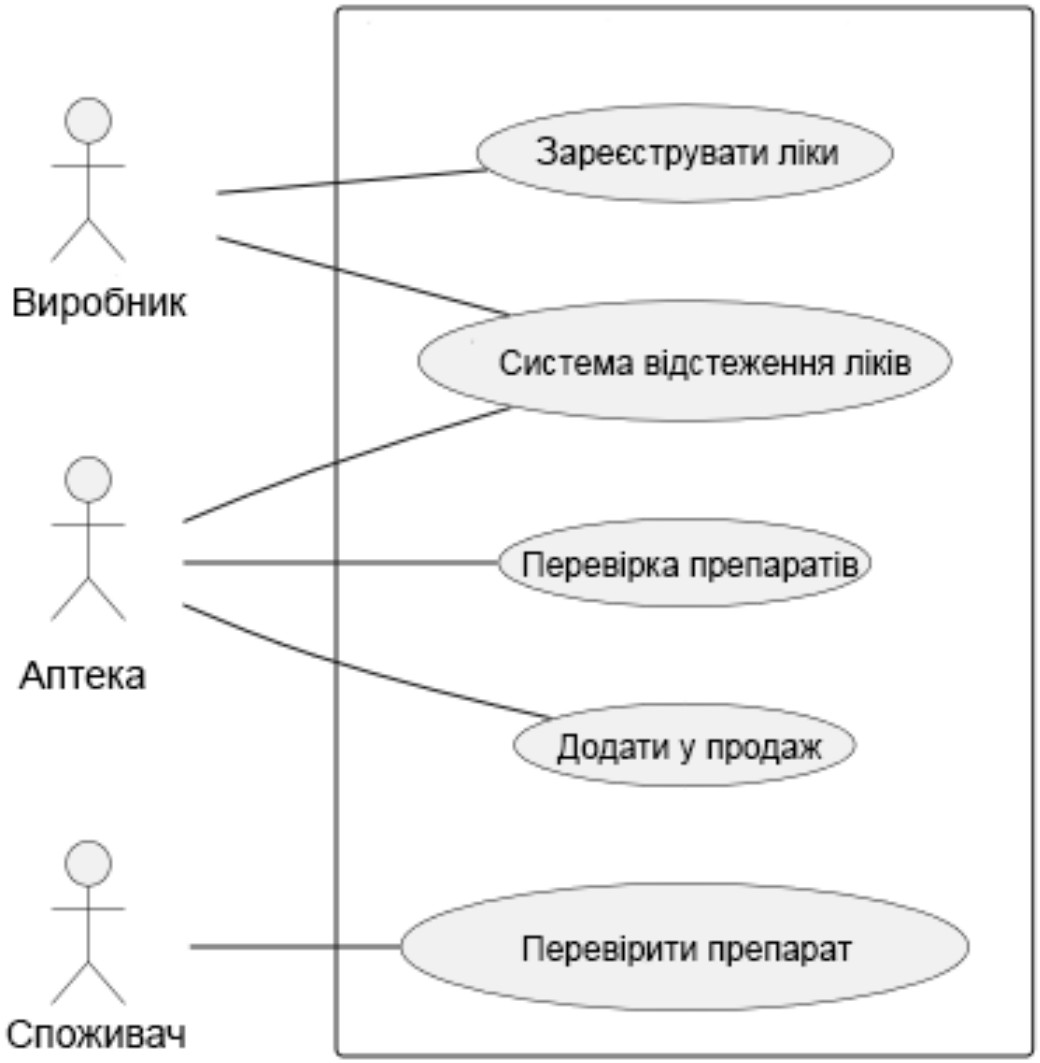


Рис. 3.1Діаграма використання

Діаграма використання ілюструє взаємодію між користувачами системи та її функціями. У системі захисту ліків від підробки та незаконного виробництва передбачені три основні ролі користувачів: Виробник, Аптека та Споживач.

Виробник здійснює реєстрацію ліків у системі, створюючи нові записи про препарати. Крім цього, виробник відповідає за використання системи відстеження ліків, яка забезпечує відстеження переміщення препаратів.

Аптека має функції перевірки препаратів, додавання їх у продаж та відстеження. Аптека отримує препарати від виробника і перевіряє їх оригінальність за допомогою. Після успішної перевірки препарати додаються до каталогу для подальшого продажу.

Споживач, у свою чергу, перевіряє оригінальність препарату за кодом препарату.

3.2 Тестування програми

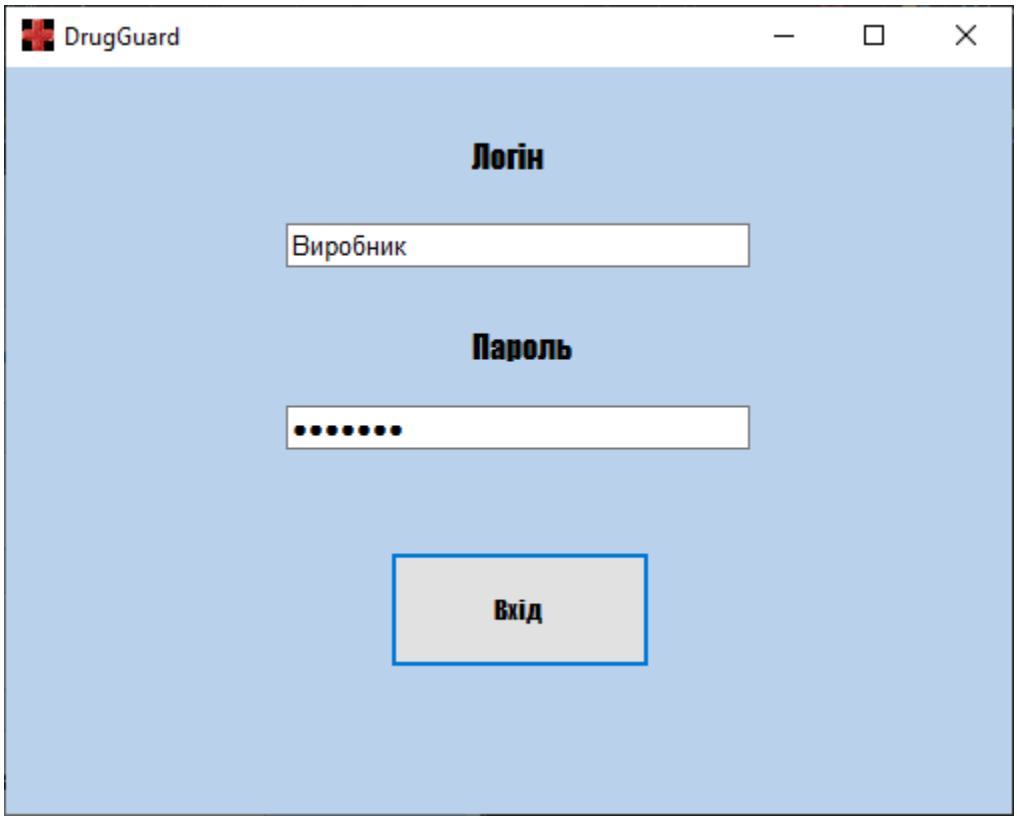


Рис. 3.2 Вікно входу

Після запуску програми вам треба буде у війти у свій профіль. Також вам буде надано первний функціонал в залежності від того увійшлі ви як виробник, аптека чи просто споживач. Також при неправильному вводі логіну або паролю, з’явиться вікно яке вам повідомить що було введено неправильно логін або пароль

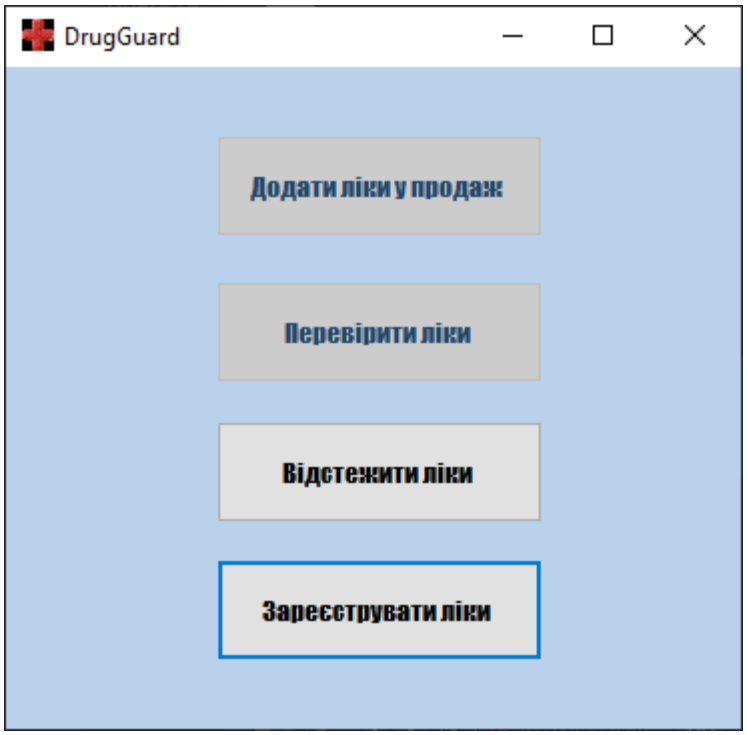


Рис. 3.3 Меню виробника

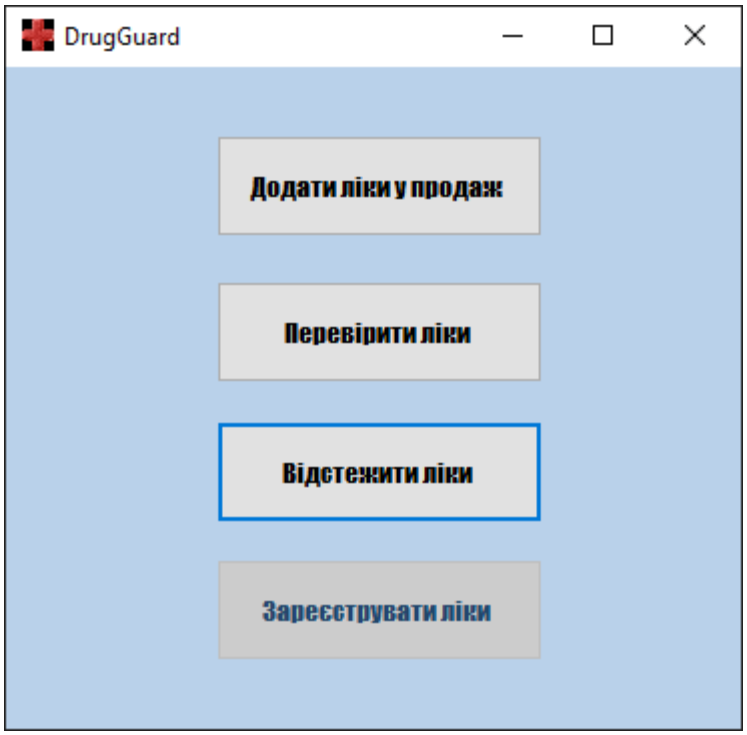


Рис. 3.4 Меню аптеки

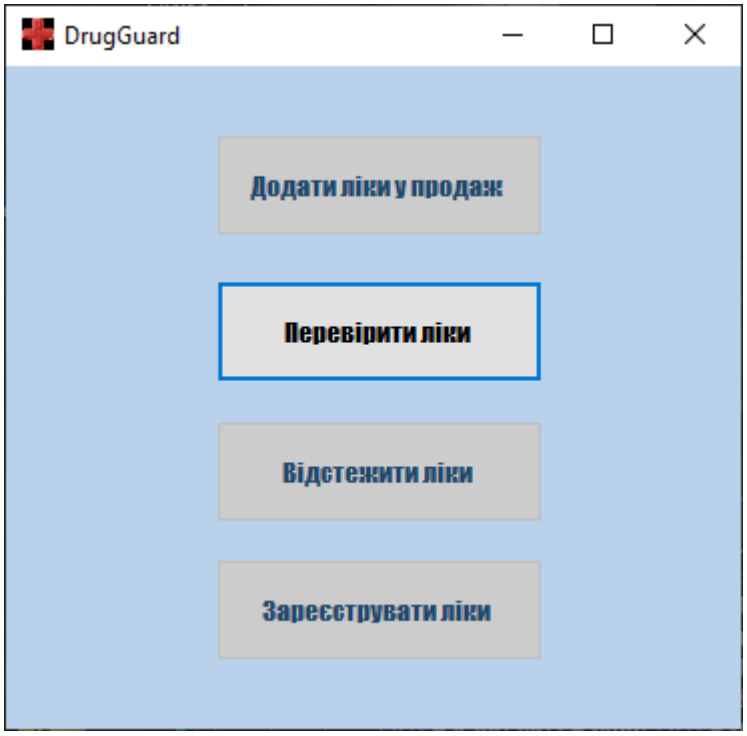


Рис. 3.5 Меню споживача

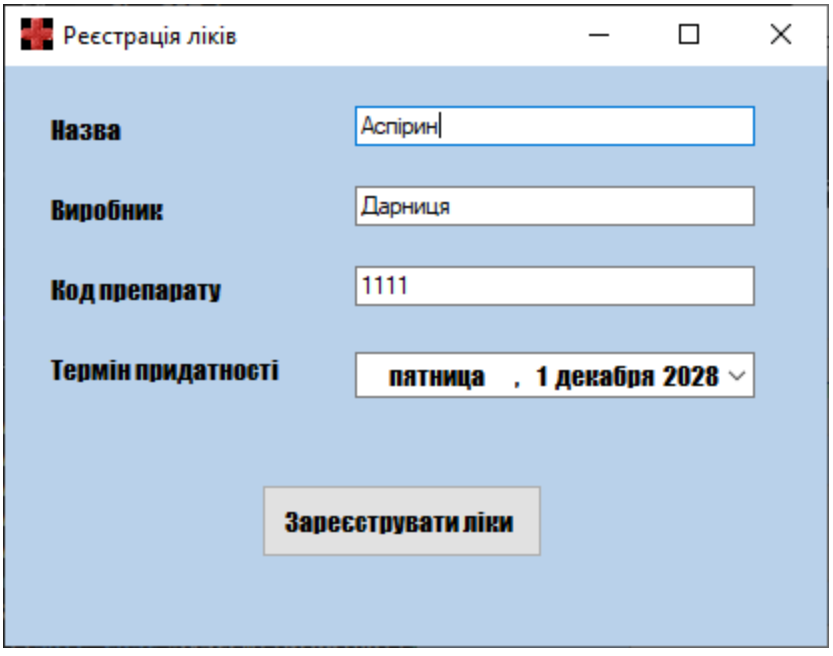


Рис. 3.6 Меню реєстрації ліків виробником

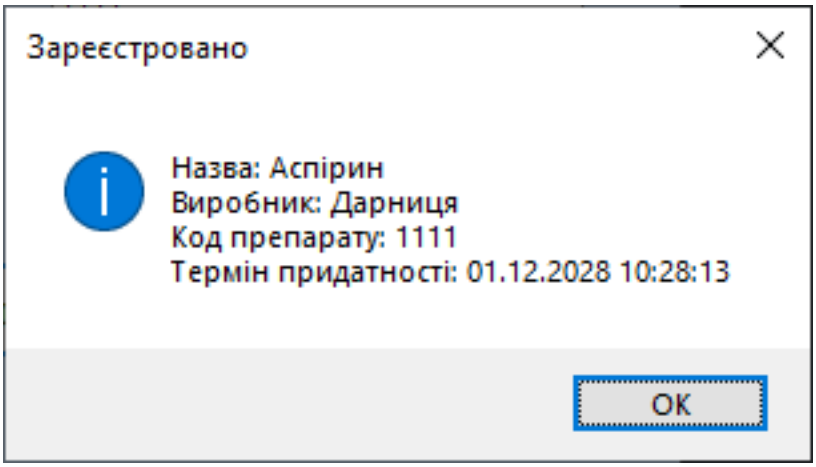


Рис. 3.7 Реєстрація препарату

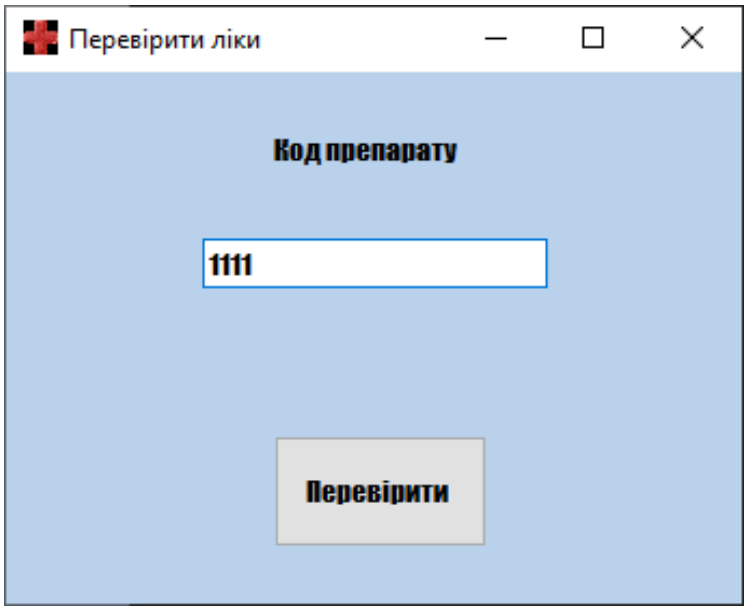


Рис. 3.8 Меню перевірки препарату за кодом

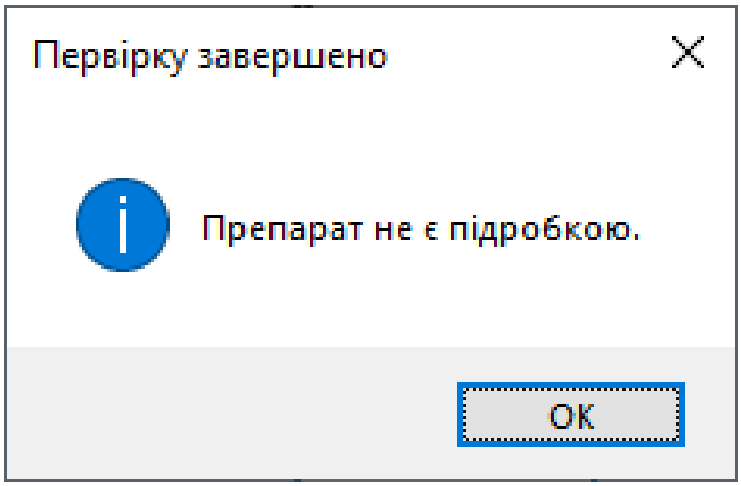


Рис. 3.9 Перевірка ліків

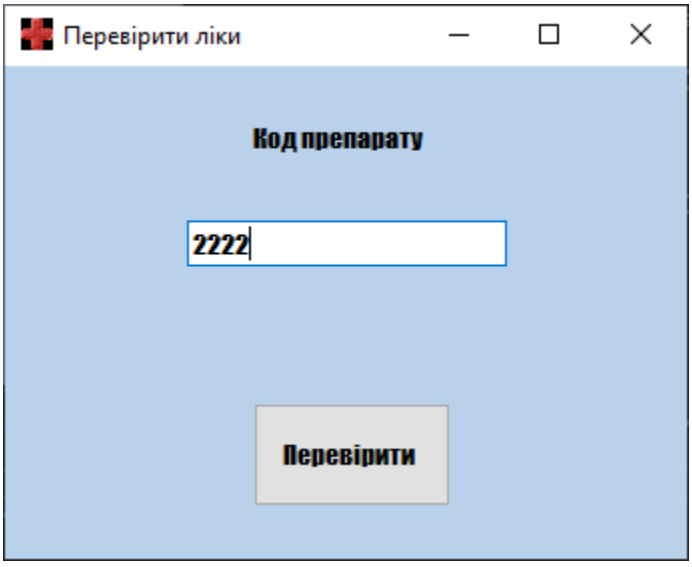


Рис. 3.10 Меню перевірки препарату за кодом

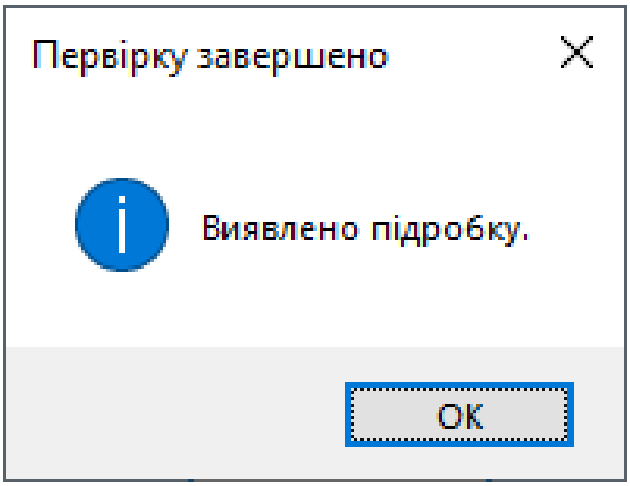


Рис. 3.11 Перевірка ліків

Таким чином, було спроектовано, розроблено і протестовано програмну систему, що дозволяє зручно і ефективно перевіряти лікарські препарати на справжність та виявляти підробки.

ВИСНОВКИ

Тема розробка програмного забезпечення для захисту ліків від підробки та незаконного виробництва на сьогоднішній день набуває надзвичайної важливості в контексті сучасних викликів у галузі охорони здоров'я. Зростання фальсифікації медичних препаратів стає серйозною загрозою для громадського здоров'я, викликаючи не тільки фінансові втрати, але й підірву довіри до медичної індустрії. В умовах глобальної пандемії, коли попит на ліки значно зрос, ця проблема стала ще актуальнішою.

В першому розділі даної роботи було проаналізовано предметну область, зокрема досліджено проблему захисту ліків від підробки і технології боротьби. Також було розглянуто протоколи транспортного рівня TCP та UDP.

У другому розділі було проаналізовано мову програмування C++ та охарактеризовано її основні особливості. Було розглянуто різні інтегровані середовища розробки програмного забезпечення мовою C++.

У третьому розділі було розроблено і протестовано програмне забезпечення для мережевої системи захисту ліків від підробки. Розроблена програма дозволяє зручно і ефективно перевіряти ліки на предмет підробки за їхнім серійним номером.

Проведений аналіз існуючих рішень виявив, що більшість методів мають певні обмеження та недоліки, які знижують їх ефективність у боротьбі з підробкою та незаконним виробництвом ліків. Ці недоліки потребують врахування під час розробки нових рішень, щоб забезпечити більш надійний захист.

На основі досліджень було визначено вимоги для створення ефективного програмного забезпечення, здатного захистити ліки від підробки та незаконного виробництва

Розроблено архітектуру програмного забезпечення для захисту ліків від підробки та незаконного виробництва. Архітектура враховує необхідність інтеграції різних технологій та забезпечує надійний механізм відстеження та контролю ліків.

Створено програмне забезпечення DrugGuard на мові C++ для захисту ліків від підробки та незаконного виробництва.

Проведено тестування програмного забезпечення для захисту ліків від підробки.

Програмне забезпечення, що забезпечує захист ліків від підробки, може забезпечити надійну систему відстеження та контролю якості лікарських засобів, зменшуючи ризик використання підроблених або низькоякісних препаратів. Така система також може сприяти попередженню та виявленню фальсифікації на ранніх стадіях, що дозволить уникнути небезпечних ситуацій для пацієнтів.

Важливість даної теми полягає також у забезпеченні безпеки глобального ланцюжка постачання медичних засобів. Інтегрована мережева система захисту може забезпечити зв'язок між виробниками, дистриб'юторами та закладами охорони здоров'я, сприяючи швидкому виявленню та реагуванню на будь-які випадки фальсифікації. Це може значно підвищити рівень довіри до медичних препаратів та забезпечити безпеку пацієнтів.

Ефективна мережева система захисту ліків від підробки є ключовим елементом боротьби зі злочинними угрупованнями, які займаються виробництвом та розповсюдженням підроблених медичних засобів. Це дозволить правоохоронним органам оперативно реагувати на випадки фальсифікації та забезпечити покарання для осіб, які порушують закони про ліцензування та безпеку медичних продуктів.

Розробка програмного забезпечення для захисту ліків від підробки може сприяти створенню міжнародних стандартів та нормативів щодо якості та безпеки

медичних препаратів. Це сприятиме створенню єдиної системи контролю якості та захисту в масштабах глобального ринку лікарських засобів.

У цілому, розробка програмного забезпечення для мережевої системи захисту ліків від підробки є важливим кроком у зміцненні сектору охорони здоров'я та захисту громадського здоров'я як на рівні країни, так і на міжнародній арені.

Таким чином, розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва має важливе значення для забезпечення якості та безпеки медичних препаратів, захисту громадського здоров'я та підвищення довіри до системи охорони здоров'я в цілому.

Результати досліджень бакалаврської роботи були представлені на Всеукраїнській науково-технічній конференції:

1. Філь І.В., Щербина І.С. Розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва мовою С++. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ. С.77-79.

ПЕРЕЛІК ПОСИЛАНЬ

1. . C++ Tutorial [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.w3schools.com/cpp/default.asp.
2. C++ Documentation [Електронний ресурс]. / – 2024. - Режим доступу до ресурсу: https://devdocs.io/cpp/
3. Coursera. What Is C++? [Електронний ресурс]. / coursera. – 2024. – Режим доступу до ресурсу: https://www.coursera.org/articles/what-is-c-plus-plus.
4. Hatice K. UML Diagrams in Software Engineering Research [Електронний ресурс] / Koc Hatice. – 2021. – Режим доступу до ресурсу: https://www.researchgate.net/publication/349973644_UML_Diagrams_in_Software_Engineering_Research_A_Systematic_Literature_Review.
5. Jammalamadaka K. Holistic research of software testing and challenges [Електронний ресурс] / K. Jammalamadaka, P. Nikhat. – 2019. – Режим доступу до ресурсу: https://doi.org/10.35940/ijitee.F1307.0486S419.
6. Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford., 2020. – 419 с.
7. SQL Server Management Studio documentation [Електронний ресурс]. / – 2024. - Режим доступу до ресурсу: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16
8. TCP/IP TCP, UDP, and IP protocols [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.ibm.com/docs/en/zos/2.4.0?topic=internets-tcpip-tcp-udp-ip-protocols.
9. Visual Studio Documentation [Електронний ресурс]. / – 2022. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022
10. Блокчейн // En.wikipedia.org. URL: https://en.wikipedia.org/wiki/Blockchain
11. Булос М. Н. К., Вілсон Дж. Т., Клаусон К. А. Геопросторовий блокчейн: обіцянки, виклики та сценарії в охороні здоров’я. С. 25.

12. ВООЗ — зростаюча загроза від фальсифікованих ліків [Електронний ресурс] / Who.int. - 2019. Режим доступу до ресурсу: https://www.who.int/bulletin/volumes/88/4/10-020410/en/

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

"Розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва мовою C++"

Виконав студент 4 курсу
групи ПД-43
Філь Ілля Владиславович
Керівник роботи
К.т.н., доц., доцент кафедри ІПЗ Щербина Ірина Сергіївна

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – покращення процесу захисту ліків від підробки та незаконного виробництва за допомогою програмного забезпечення мережевої системи DrugGuard мовою C++

Об'єкт дослідження – процес захисту від підробки та незаконного виробництва ліків

Предмет дослідження – програмне забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз існуючих рішень для захисту лікарських препаратів від підробки та незаконного виробництва.
2. Створення функціональних та нефункціональних вимог до програмного забезпечення для захисту ліків.
3. Розробка архітектури програмного забезпечення для захисту ліків від підробки та незаконного виробництва.
4. Розробка програмного забезпечення для мережевої системи для захисту ліків від підробки та незаконного виробництва.
5. Тестування програмного забезпечення для захисту ліків від підробки та незаконного виробництва.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	TraceLink	AxiCorp	DrugGuard
Відстеження ланцюжка поставок	+	-	+
Всі препарати мають унікальні ідентифікатори	+	-	+
Підтримка Windows	-	+	+
Підтримка звітності	-	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Забезпечення ідентифікації лікарських препаратів на всіх етапах ланцюжка поставок.
2. Відстеження та моніторинг шляху препаратів від виробника до споживача.
3. Верифікація автентичності препаратів на основі ідентифікаторів та даних з реєстру.
4. Виявлення випадків підробки.
5. Генерація звітів про рух препаратів.

Нефункціональні вимоги:

1. Підтримка Windows.
2. Локалізація українською мовою.

5

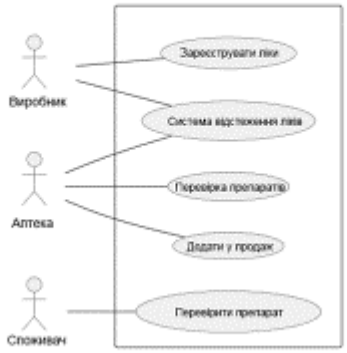
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Visual Studio

SQL Server management studio

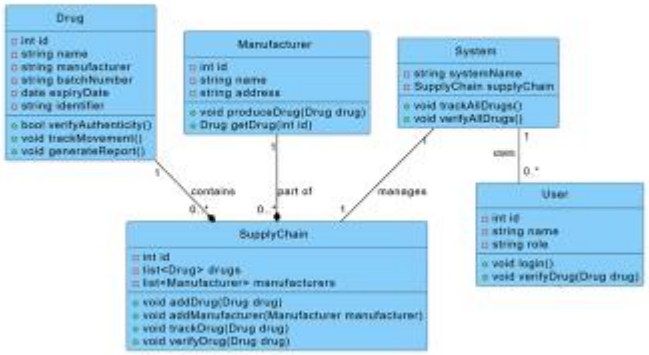
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



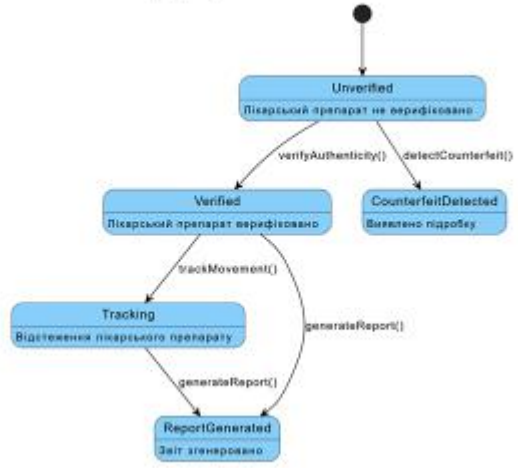
7

Діаграма класів



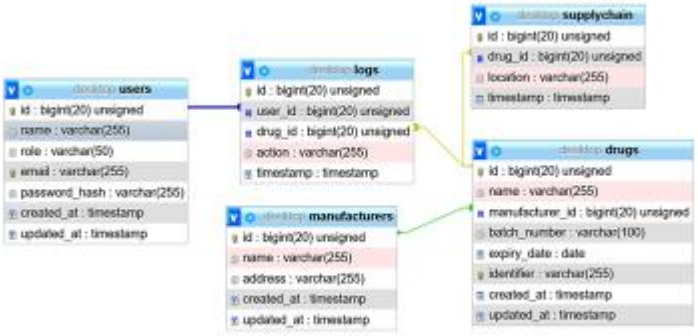
8

Діаграма станів



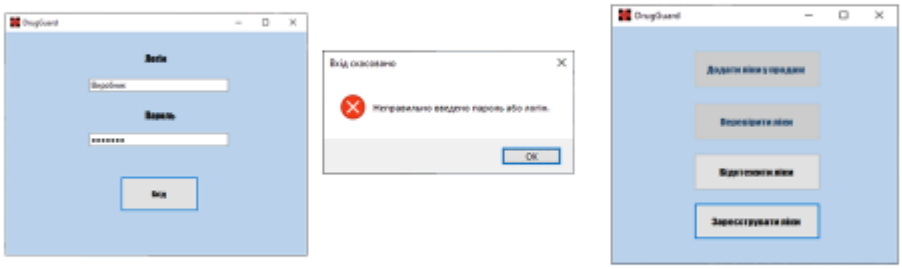
9

Схема бази даних



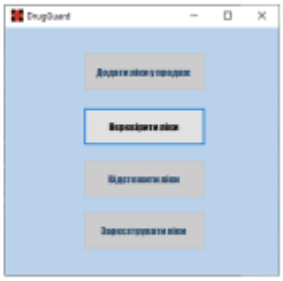
10

Екранні форми



11

Екранні форми

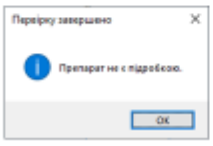


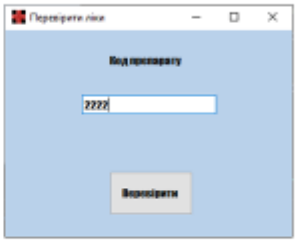
13

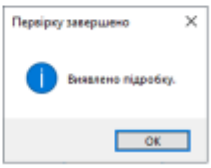
Екранні форми

12

Екранні форми

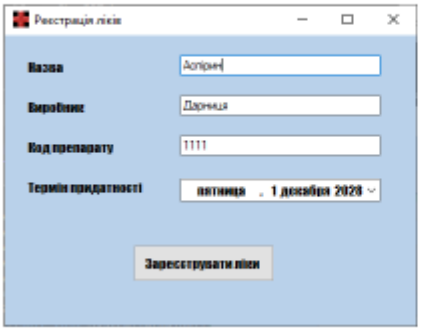


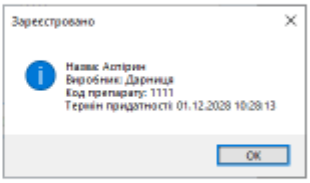




14

Екранні форми





15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1.Філь І.В., Щербина І.С. Розробка програмного забезпечення мережевої системи для захисту ліків від підробки та незаконного виробництва мовою с++. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ». Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ(ст. 77-79)

14

ВИСНОВКИ

1. Проведений аналіз існуючих рішень виявив, що більшість методів мають певні обмеження та недоліки, які знижують їх ефективність у боротьбі з підробкою та незаконним виробництвом ліків.
2. Досліджено вимоги для створення ефективного програмного забезпечення для захисту ліків на всіх етапах ланцюжка поставок.
3. Розроблено архітектуру програмного забезпечення для захисту ліків від підробки та незаконного виробництва
4. Створено програмне забезпечення DrugGuard на мові С++ з усіма необхідними функціями для захисту ліків.
5. Проведено тестування програмного забезпечення для захисту ліків від підробки.

15

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

AddMedicineForSaleForm.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;

    public ref class CheckMedicineForm : public System::Windows::Forms::Form
    {
    public:
        CheckMedicineForm(Form^ previousForm)
        {
            InitializeComponent();
            this->previousForm = previousForm;
        }

    protected:
        ~CheckMedicineForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::TextBox^ txtCode;
    private: System::Windows::Forms::Button^ btnCheck;
    private: System::Windows::Forms::Label^ lblResult;
    private: System::Windows::Forms::Button^ btnBack;
    private: Form^ previousForm;
    private: System::Windows::Forms::Label^ label1;

    private:
        System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
            this->txtCode = (gcnew System::Windows::Forms::TextBox());
            this->btnCheck = (gcnew System::Windows::Forms::Button());
            this->lblResult = (gcnew System::Windows::Forms::Label());
            this->btnBack = (gcnew System::Windows::Forms::Button());
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // txtCode
            //
            this->txtCode->Location = System::Drawing::Point(294, 223);
            this->txtCode->Multiline = true;
            this->txtCode->Name = L"txtCode";
            this->txtCode->Size = System::Drawing::Size(210, 38);
            this->txtCode->TabIndex = 0;
            this->txtCode->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;
            //
            // btnCheck
            //
            this->btnCheck->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnCheck->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnCheck->ForeColor = System::Drawing::SystemColors::Control;
            this->btnCheck->Location = System::Drawing::Point(154, 223);
            this->btnCheck->Name = L"btnCheck";
            this->btnCheck->Size = System::Drawing::Size(108, 38);
            this->btnCheck->TabIndex = 1;
            this->btnCheck->Text = L"Перевірити";
            this->btnCheck->UseVisualStyleBackColor = false;
            this->btnCheck->Click += gcnew System::EventHandler(this, &CheckMedicineForm::btnCheck_Click);
            //
            // lblResult
```

```
            //
            this->lblResult->AutoSize = true;
            this->lblResult->Location = System::Drawing::Point(154, 280);
            this->lblResult->MaximumSize = System::Drawing::Size(600, 0);
            this->lblResult->Name = L"lblResult";
            this->lblResult->Size = System::Drawing::Size(0, 21);
            this->lblResult->TabIndex = 2;
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(24, 25);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(182, 52);
            this->btnBack->TabIndex = 3;
            this->btnBack->Text = L"<-- Повернутися назад";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &CheckMedicineForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
            this->label1->Location = System::Drawing::Point(304, 130);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(188, 45);
            this->label1->TabIndex = 5;
            this->label1->Text = L"DrugGuard";
            //
            // CheckMedicineForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->lblResult);
            this->Controls->Add(this->btnCheck);
            this->Controls->Add(this->txtCode);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"CheckMedicineForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void btnCheck_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ code = txtCode->Text;

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT * FROM Medicines WHERE Code=@code", conn);

        cmd->Parameters->AddWithValue("@code", code);

        SqlDataReader^ reader;

        try {
            conn->Open();
            reader = cmd->ExecuteReader();

            if (reader->Read()) {
                String^ name = reader["Name"]->ToString();
                String^ manufactureDate = reader["ManufactureDate"]->ToString();
                String^ expiryDate = reader["ExpiryDate"]->ToString();

                lblResult->Text = "Назва препарату: " + name + "\nКод препарату: " + code + "\nДата виробництва: " + manufactureDate + "\nТермін придатності: " + expiryDate + "\nЛіки не є підробкою";
            }
            else {
                lblResult->Text = "Виявлено підробку";
            }
```

```
            }
            catch (Exception^ ex) {
                MessageBox::Show("Помилка
підключення до бази даних: " + ex->Message);
            }
            finally {
                if (reader != nullptr) {
                    reader->Close();
                }
                conn->Close();
        }
            }
        }

        private:                System::Void
btnBack_Click(System::Object^           sender,
System::EventArgs^ e) {
            this->Close();
            previousForm->Show();
        }
        };
```

CheckMedicineForm.h

```
        #pragma once

        namespace MedicationProtectionApp {

            using namespace System;
            using                    namespace
System::ComponentModel;
            using namespace System::Collections;
            using namespace System::Windows::Forms;
            using namespace System::Data;
            using namespace System::Drawing;
            using namespace System::Data::SqlClient;

            public ref class CheckMedicineForm : public
System::Windows::Forms::Form
            {
            public:
                CheckMedicineForm(Form^
previousForm)
                {
                    InitializeComponent();
                    this->previousForm = previousForm;
                }

            protected:
                ~CheckMedicineForm()
                {
                    if (components)
                    {
                        delete components;
                    }
                }

            private:
System::Windows::Forms::TextBox^ txtCode;
            private:  System::Windows::Forms::Button^
btnCheck;
            private:    System::Windows::Forms::Label^
lblResult;
            private:  System::Windows::Forms::Button^
btnBack;
            private: Form^ previousForm;
            private:    System::Windows::Forms::Label^
label1;

            private:
                System::ComponentModel::Container^
components;

            #pragma region Windows Form Designer
generated code
                void InitializeComponent(void)
                {
                    this->txtCode          =          (gcnew
System::Windows::Forms::TextBox());
                    this->btnCheck          =          (gcnew
System::Windows::Forms::Button());
                    this->lblResult          =          (gcnew
System::Windows::Forms::Label());
                    this->btnBack          =          (gcnew
System::Windows::Forms::Button());
                    this->label1          =          (gcnew
System::Windows::Forms::Label());
                    this->SuspendLayout();
                    //
                    // txtCode
                    //
                    this->txtCode->Location                =
System::Drawing::Point(294, 223);
                    this->txtCode->Multiline = true;
                    this->txtCode->Name = L"txtCode";
                    this->txtCode->Size                    =
System::Drawing::Size(210, 38);
                    this->txtCode->TabIndex = 0;
                    this->txtCode->TextAlign               =
System::Windows::Forms::HorizontalAlignment::Cente
r;
                    //
                    // btnCheck
                    //
                    this->btnCheck->BackColor              =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->btnCheck->FlatStyle              =
System::Windows::Forms::FlatStyle::Popup;
```

```
            this->btnCheck->ForeColor = System::Drawing::SystemColors::Control;
            this->btnCheck->Location = System::Drawing::Point(154, 223);
            this->btnCheck->Name = L"btnCheck";
            this->btnCheck->Size = System::Drawing::Size(108, 38);
            this->btnCheck->TabIndex = 1;
            this->btnCheck->Text = L"Перевірити";
            this->btnCheck->UseVisualStyleBackColor = false;
            this->btnCheck->Click += gcnew System::EventHandler(this, &CheckMedicineForm::btnCheck_Click);
            //
            // lblResult
            //
            this->lblResult->AutoSize = true;
            this->lblResult->Location = System::Drawing::Point(154, 280);
            this->lblResult->MaximumSize = System::Drawing::Size(600, 0);
            this->lblResult->Name = L"lblResult";
            this->lblResult->Size = System::Drawing::Size(0, 21);
            this->lblResult->TabIndex = 2;
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(24, 25);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(182, 52);
            this->btnBack->TabIndex = 3;
            this->btnBack->Text = L"<-- Повернутися назад";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &CheckMedicineForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
            this->label1->Location = System::Drawing::Point(304, 130);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(188, 45);
            this->label1->TabIndex = 5;
            this->label1->Text = L"DrugGuard";
            //
            // CheckMedicineForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->lblResult);
            this->Controls->Add(this->btnCheck);
            this->Controls->Add(this->txtCode);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"CheckMedicineForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void btnCheck_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ code = txtCode->Text;

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT * FROM Medicines WHERE Code=@code", conn);

        cmd->Parameters->AddWithValue("@code", code);

        SqlDataReader^ reader;

        try {
            conn->Open();
```

```
                reader = cmd->ExecuteReader();

                if (reader->Read()) {
                    String^ name = reader["Name"]->ToString();
                    String^ manufactureDate = reader["ManufactureDate"]->ToString();
                    String^ expiryDate = reader["ExpiryDate"]->ToString();

                    lblResult->Text = "Назва препарату: " + name + "\nКод препарату: " + code + "\nДата виробництва: " + manufactureDate + "\nТермін придатності: " + expiryDate + "\nЛіки не є підробкою";
                }
                else {
                    lblResult->Text = "Виявлено підробку";
                }
            };
        }
                }
                catch (Exception^ ex) {
                    MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
                }
                finally {
                    if (reader != nullptr) {
                        reader->Close();
                    }
                    conn->Close();
                }
            }

        private: System::Void btnBack_Click(System::Object^ sender, System::EventArgs^ e) {
            this->Close();
            previousForm->Show();
        }
```

ConsumerForm.h

```
        #pragma once

        #include "CheckMedicineForm.h"

        namespace MedicationProtectionApp {

            using namespace System;
            using namespace System::ComponentModel;
            using namespace System::Collections;
            using namespace System::Windows::Forms;
            using namespace System::Data;
            using namespace System::Drawing;

            public ref class ConsumerForm : public System::Windows::Forms::Form
            {
            public:
                ConsumerForm(String^ username, Form^ loginForm)
                {
                    InitializeComponent();
                    lblWelcome->Text = "Вітаю, " + username + ", ви авторизувались як споживач";
                    lblWelcome->Font = gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Bold);
                    CenterLabel(lblWelcome);
                    this->username = username;
                    previousForm = loginForm;
                }
            protected:
                ~ConsumerForm()
                {
                    if (components)
                    {
                        delete components;
                    }
                }

            private: System::Windows::Forms::Label^ lblWelcome;
            private: System::Windows::Forms::Button^ btnCheckMedicine;
            private: System::Windows::Forms::Button^ btnBack;
            private: Form^ previousForm;
            private: String^ username;
            private: System::Windows::Forms::Label^ label1;


            private:
                System::ComponentModel::Container^ components;

        #pragma region Windows Form Designer generated code
                void InitializeComponent(void)
                {
```

```
            this->lblWelcome = (gcnew
System::Windows::Forms::Label());
            this->btnCheckMedicine = (gcnew
System::Windows::Forms::Button());
            this->btnBack = (gcnew
System::Windows::Forms::Button());
            this->label1 = (gcnew
System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // lblWelcome
            //
            this->lblWelcome->AutoSize = true;
            this->lblWelcome->Location =
System::Drawing::Point(0, 20);
            this->lblWelcome->Name =
L"lblWelcome";
            this->lblWelcome->Size =
System::Drawing::Size(0, 21);
            this->lblWelcome->TabIndex = 0;
            //
            // btnCheckMedicine
            //
            this->btnCheckMedicine->BackColor
= System::Drawing::SystemColors::ActiveCaptionText;
            this->btnCheckMedicine->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
            this->btnCheckMedicine->ForeColor =
System::Drawing::SystemColors::Control;
            this->btnCheckMedicine->Location =
System::Drawing::Point(306, 283);
            this->btnCheckMedicine->Name =
L"btnCheckMedicine";
            this->btnCheckMedicine->Size =
System::Drawing::Size(204, 38);
            this->btnCheckMedicine->TabIndex =
1;
            this->btnCheckMedicine->Text =
L"Перевірити ліки";
            this->btnCheckMedicine-
>UseVisualStyleBackColor = false;
            this->btnCheckMedicine->Click +=
gcnew System::EventHandler(this,
&ConsumerForm::btnCheckMedicine_Click);
            //
            // btnBack
            //
            this->btnBack->BackColor =
System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor =
System::Drawing::SystemColors::Control;
            this->btnBack->Location =
System::Drawing::Point(306, 394);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size =
System::Drawing::Size(204, 38);
            this->btnBack->TabIndex = 2;
            this->btnBack->Text = L"<--
Повернутися назад";
            this->btnBack-
>UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew
System::EventHandler(this,
&ConsumerForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->FlatStyle =
System::Windows::Forms::FlatStyle::Flat;
            this->label1->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 24,
System::Drawing::FontStyle::Bold));
            this->label1->Location =
System::Drawing::Point(309, 102);
            this->label1->Name = L"label1";
            this->label1->Size =
System::Drawing::Size(188, 45);
            this->label1->TabIndex = 5;
            this->label1->Text = L"DrugGuard";
            //
            // ConsumerForm
            //
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize =
System::Drawing::Size(796, 521);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this-
>btnCheckMedicine);
            this->Controls->Add(this-
>lblWelcome);
            this->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedToo
lWindow;
            this->Name = L"ConsumerForm";
            this->Text = L"DrugGuard -
споживач";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

        private: System::Void
btnCheckMedicine_Click(System::Object^ sender,
System::EventArgs^ e) {
```

```
            CheckMedicineForm^ checkForm = gcnew CheckMedicineForm(this);
            checkForm->ShowDialog();
        }

        private: System::Void btnBack_Click(System::Object^ sender, System::EventArgs^ e) {
            this->Close();
            previousForm->Show();
        }

        private: void CenterLabel(System::Windows::Forms::Label^ label)
        {
            label->Location = System::Drawing::Point((this->ClientSize.Width - label->Width) / 2, label->Location.Y);
        }
        };
    }
```

GenerateReportForSaleForm.h

```
    #pragma once

    #include "ReportForm.h"
    #include "KeyValuePair.h"

    namespace MedicationProtectionApp {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace System::Data::SqlClient;

        public ref class GenerateReportForSaleForm : public System::Windows::Forms::Form
        {
        public:
            GenerateReportForSaleForm(Form^ previousForm)
            {
                InitializeComponent();
                this->previousForm = previousForm;
                LoadMedicinesForSale();
            }

        protected:
            ~GenerateReportForSaleForm()
            {
                if (components)
                {
                    delete components;
                }
            }

        private: System::Windows::Forms::ComboBox^ cmbMedicines;
        private: System::Windows::Forms::Button^ btnGenerateReport;
        private: System::Windows::Forms::Label^ lblMessage;
        private: System::Windows::Forms::Button^ btnBack;

        private:
            Form^ previousForm;
        private: System::Windows::Forms::Label^ label1;
        private: System::Windows::Forms::Label^ label4;

            System::ComponentModel::Container^ components;

    #pragma region Windows Form Designer generated code
            void InitializeComponent(void)
            {
                this->cmbMedicines = (gcnew System::Windows::Forms::ComboBox());
                this->btnGenerateReport = (gcnew System::Windows::Forms::Button());
                this->lblMessage = (gcnew System::Windows::Forms::Label());
                this->btnBack = (gcnew System::Windows::Forms::Button());
                this->label1 = (gcnew System::Windows::Forms::Label());
                this->label4 = (gcnew System::Windows::Forms::Label());
                this->SuspendLayout();
                //
                // cmbMedicines
                //
                this->cmbMedicines->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;
                this->cmbMedicines->FormattingEnabled = true;
```

```
            this->cmbMedicines->Location = System::Drawing::Point(294, 212);
            this->cmbMedicines->Name = L"cmbMedicines";
            this->cmbMedicines->Size = System::Drawing::Size(200, 25);
            this->cmbMedicines->TabIndex = 0;
            //
            // btnGenerateReport
            //
            this->btnGenerateReport->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnGenerateReport->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnGenerateReport->Font = (gcnew System::Drawing::Font(L"Segoe UI Semibold", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->btnGenerateReport->ForeColor = System::Drawing::SystemColors::Control;
            this->btnGenerateReport->Location = System::Drawing::Point(294, 302);
            this->btnGenerateReport->Name = L"btnGenerateReport";
            this->btnGenerateReport->Size = System::Drawing::Size(200, 31);
            this->btnGenerateReport->TabIndex = 1;
            this->btnGenerateReport->Text = L"Генерувати звіт";
            this->btnGenerateReport->UseVisualStyleBackColor = false;
            this->btnGenerateReport->Click += gcnew System::EventHandler(this, &GenerateReportForSaleForm::btnGenerateReport_Click);
            //
            // lblMessage
            //
            this->lblMessage->AutoSize = true;
            this->lblMessage->Location = System::Drawing::Point(90, 110);
            this->lblMessage->Name = L"lblMessage";
            this->lblMessage->Size = System::Drawing::Size(0, 19);
            this->lblMessage->TabIndex = 2;
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(12, 12);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(200, 30);
            this->btnBack->TabIndex = 3;
            this->btnBack->Text = L"<-- Повернутися назад";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &GenerateReportForSaleForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
            this->label1->Location = System::Drawing::Point(306, 101);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(188, 45);
            this->label1->TabIndex = 5;
            this->label1->Text = L"DrugGuard";
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label4->Font = (gcnew System::Drawing::Font(L"Segoe UI", 16, System::Drawing::FontStyle::Bold));
            this->label4->Location = System::Drawing::Point(289, 170);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(214, 30);
            this->label4->TabIndex = 8;
            this->label4->Text = L"Виберіть препарат";
            //
            // GenerateReportForSaleForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->lblMessage);
```

```
            this->Controls->Add(this->btnGenerateReport);
            this->Controls->Add(this->cmbMedicines);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 10));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"GenerateReportForSaleForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void LoadMedicinesForSale()
    {
        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT Code, Name FROM MedicinesForSale", conn);
        SqlDataReader^ reader;

        try {
            conn->Open();
            reader = cmd->ExecuteReader();

            while (reader->Read()) {
                String^ code = reader["Code"]->ToString();
                String^ name = reader["Name"]->ToString();
                cmbMedicines->Items->Add(gcnew KeyValuePair(code, name));
            }
        }
        catch (Exception^ ex) {
            MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
        }
        finally {
            if (reader != nullptr) {
                reader->Close();
            }
            conn->Close();
        }
    }

    private: System::Void btnGenerateReport_Click(System::Object^ sender, System::EventArgs^ e) {
        if (cmbMedicines->SelectedItem == nullptr) {
            lblMessage->Text = "Будь ласка, виберіть препарат зі списку.";
            return;
        }

        KeyValuePair^ selectedPair = (KeyValuePair^)cmbMedicines->SelectedItem;
        String^ code = selectedPair->Key;

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT Name, Code, Manufacturer, ManufactureDate, ExpiryDate, Description, Price, FilePath FROM MedicinesForSale WHERE Code=@code", conn);

        cmd->Parameters->AddWithValue("@code", code);

        SqlDataReader^ reader;

        try {
            conn->Open();
            reader = cmd->ExecuteReader();

            if (reader->Read()) {
                String^ name = reader["Name"]->ToString();
                String^ manufacturer = reader["Manufacturer"]->ToString();
                String^ manufactureDate = reader["ManufactureDate"]->ToString();
                String^ expiryDate = reader["ExpiryDate"]->ToString();
                String^ description = reader["Description"]->ToString();
                String^ price = reader["Price"]->ToString();
                String^ filePath = reader["FilePath"]->ToString();

                String^ report = "Звіт по препарату \"" + name + "\" успішно згенеровано!\n\n" +
                    "Назва препарату: " + name + "\n" +
                    "Код препарату: " + code + "\n" +
                    "Виробник: " + manufacturer + "\n" +
                    "Дата виробництва: " + manufactureDate + "\n" +
                    "Термін придатності: " + expiryDate + "\n" +
                    "Опис: " + description + "\n" +
```

```
                    "Ціна: " + price;

                ReportForm^ reportForm = gcnew
ReportForm(report, filePath);
                reportForm->ShowDialog();
            }
            else {
                lblMessage->Text = "Ліки не
знайдено.";
            }
        }
        catch (Exception^ ex) {
            lblMessage->Text = "Помилка
підключення до бази даних: " + ex->Message;
        }
        finally {
            if (reader != nullptr) {
                reader->Close();
            }
            conn->Close();
        }
    }

    private: System::Void
btnBack_Click(System::Object^ sender,
System::EventArgs^ e) {
        this->Close();
        previousForm->Show();
    }
    };
}
```

KeyValuePair.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;

    public ref class KeyValuePair
    {
    public:
        String^ Key;
        String^ Value;

        KeyValuePair(String^ key, String^ value)
        {
            this->Key = key;
            this->Value = value;
        }

        virtual String^ ToString() override
        {
            return this->Value;
        }
    };
}
```

LoginForm.h

```
#pragma once

#include "RegisterForm.h"
#include "ManufacturerMenuForm.h"
#include "PharmacyMenuForm.h"
#include "ConsumerForm.h"

namespace MedicationProtectionApp {

    using namespace System;
    using namespace
System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;

    public ref class LoginForm : public
System::Windows::Forms::Form
    {
    public:
        LoginForm(void)
        {
            InitializeComponent();
        }

    protected:
        ~LoginForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private:
System::Windows::Forms::TextBox^ txtLogin;
    private:
System::Windows::Forms::TextBox^ txtPassword;
    private: System::Windows::Forms::Button^
btnLogin;
    private: System::Windows::Forms::Button^
btnRegister;
```

```
        private: System::Windows::Forms::Label^
label1;
        private: System::Windows::Forms::Label^
label2;
        private: System::Windows::Forms::Label^
label3;
        private: System::Windows::Forms::Label^
label4;


        private:
          System::ComponentModel::Container^
components;

    #pragma region Windows Form Designer
generated code
            void InitializeComponent(void)
            {
              this->txtLogin = (gcnew
System::Windows::Forms::TextBox());
              this->txtPassword = (gcnew
System::Windows::Forms::TextBox());
              this->btnLogin = (gcnew
System::Windows::Forms::Button());
              this->btnRegister = (gcnew
System::Windows::Forms::Button());
              this->label1 = (gcnew
System::Windows::Forms::Label());
              this->label2 = (gcnew
System::Windows::Forms::Label());
              this->label3 = (gcnew
System::Windows::Forms::Label());
              this->label4 = (gcnew
System::Windows::Forms::Label());
              this->SuspendLayout();
              //
              // txtLogin
              //
              this->txtLogin->Location =
System::Drawing::Point(306, 247);
              this->txtLogin->Name = L"txtLogin";
              this->txtLogin->Size =
System::Drawing::Size(200, 29);
              this->txtLogin->TabIndex = 0;
              //
              // txtPassword
              //
              this->txtPassword->Location =
System::Drawing::Point(306, 318);
              this->txtPassword->Name =
L"txtPassword";
              this->txtPassword->PasswordChar = '•';
              this->txtPassword->Size =
System::Drawing::Size(200, 29);
              this->txtPassword->TabIndex = 1;
              this->txtPassword-
>UseSystemPasswordChar = true;
              //
              // btnLogin
              //
              this->btnLogin->BackColor =
System::Drawing::SystemColors::ActiveCaptionText;
              this->btnLogin->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
              this->btnLogin->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12));
              this->btnLogin->ForeColor =
System::Drawing::SystemColors::Control;
              this->btnLogin->Location =
System::Drawing::Point(219, 378);
              this->btnLogin->Name = L"btnLogin";
              this->btnLogin->Size =
System::Drawing::Size(146, 54);
              this->btnLogin->TabIndex = 2;
              this->btnLogin->Text = L"Увійти";
              this->btnLogin-
>UseVisualStyleBackColor = false;
              this->btnLogin->Click += gcnew
System::EventHandler(this,
&LoginForm::btnLogin_Click);
              //
              // btnRegister
              //
              this->btnRegister->BackColor =
System::Drawing::SystemColors::ActiveCaptionText;
              this->btnRegister->FlatStyle =
System::Windows::Forms::FlatStyle::Popup;
              this->btnRegister->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12));
              this->btnRegister->ForeColor =
System::Drawing::SystemColors::Control;
              this->btnRegister->Location =
System::Drawing::Point(444, 378);
              this->btnRegister->Name =
L"btnRegister";
              this->btnRegister->RightToLeft =
System::Windows::Forms::RightToLeft::No;
              this->btnRegister->Size =
System::Drawing::Size(146, 54);
              this->btnRegister->TabIndex = 3;
              this->btnRegister->Text =
L"Зареєструватися";
              this->btnRegister-
>UseCompatibleTextRendering = true;
              this->btnRegister-
>UseVisualStyleBackColor = false;
              this->btnRegister->Click += gcnew
System::EventHandler(this,
&LoginForm::btnRegister_Click);
              //
              // label1
              //
              this->label1->AutoSize = true;
              this->label1->FlatStyle =
System::Windows::Forms::FlatStyle::Flat;
```

```
            this->label1->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 24,
System::Drawing::FontStyle::Bold));
            this->label1->Location =
System::Drawing::Point(309, 102);
            this->label1->Name = L"label1";
            this->label1->Size =
System::Drawing::Size(188, 45);
            this->label1->TabIndex = 4;
            this->label1->Text = L"DrugGuard";
            this->label1->Click += gcnew
System::EventHandler(this,
&LoginForm::label1_Click);
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label2->Location =
System::Drawing::Point(302, 223);
            this->label2->Name = L"label2";
            this->label2->Size =
System::Drawing::Size(49, 21);
            this->label2->TabIndex = 5;
            this->label2->Text = L"Логін";
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->BackColor =
System::Drawing::SystemColors::Control;
            this->label3->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label3->Location =
System::Drawing::Point(302, 294);
            this->label3->Name = L"label3";
            this->label3->Size =
System::Drawing::Size(63, 21);
            this->label3->TabIndex = 6;
            this->label3->Text = L"Пароль";
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->FlatStyle =
System::Windows::Forms::FlatStyle::Flat;
            this->label4->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 16,
System::Drawing::FontStyle::Bold));
            this->label4->Location =
System::Drawing::Point(244, 170);
            this->label4->Name = L"label4";
            this->label4->Size =
System::Drawing::Size(315, 30);
            this->label4->TabIndex = 7;
            this->label4->Text = L"Авторизуйтесь
у застосунку";
            this->label4->Click += gcnew
System::EventHandler(this,
&LoginForm::label4_Click);
            //
            // LoginForm
            //
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::None;
            this->BackColor =
System::Drawing::SystemColors::Control;
            this->ClientSize =
System::Drawing::Size(796, 521);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this-
>btnRegister);
            this->Controls->Add(this->btnLogin);
            this->Controls->Add(this-
>txtPassword);
            this->Controls->Add(this->txtLogin);
            this->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedToo
lWindow;
            this->Name = L"LoginForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

        private: System::Void
btnLogin_Click(System::Object^ sender,
System::EventArgs^ e) {
            String^ login = txtLogin->Text;
            String^ password = txtPassword->Text;

            SqlConnection^ conn = gcnew
SqlConnection("Server=DESKTOP-
ACME1K0;Database=MedicationSystem;Integrated
Security=True;");
            SqlCommand^ cmd = gcnew
SqlCommand("SELECT * FROM Users WHERE
Login=@login AND Password=@password", conn);
```

```
            cmd->Parameters->AddWithValue("@login", login);
            cmd->Parameters->AddWithValue("@password", password);

            SqlDataReader^ reader;

            try {
                conn->Open();
                reader = cmd->ExecuteReader();

                if (reader->Read()) {
                    String^ role = reader["Role"]->ToString();
                    String^ username = reader["Login"]->ToString();
                    this->Hide();
                    if (role == "Виробник") {
                        ManufacturerMenuForm^ menuForm = gcnew ManufacturerMenuForm(username, this);
                        menuForm->ShowDialog();
                    }
                    else if (role == "Аптека") {
                        PharmacyMenuForm^ menuForm = gcnew PharmacyMenuForm(username, this);
                        menuForm->ShowDialog();
                    }
                    else if (role == "Споживач") {
                        ConsumerForm^ cf = gcnew ConsumerForm(username, this);
                        cf->ShowDialog();
                    }
                    this->Show();
                }
                else {
                    MessageBox::Show("Невірний логін або пароль");
                }
            }
            catch (Exception^ ex) {
                MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
            }
            finally {
                if (reader != nullptr) {
                    reader->Close();
                }
                conn->Close();
            }
        }

        private: System::Void btnRegister_Click(System::Object^ sender, System::EventArgs^ e) {
            RegisterForm^ rf = gcnew RegisterForm();
            rf->ShowDialog();
        }
        private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
        }
        private: System::Void label4_Click(System::Object^ sender, System::EventArgs^ e) {
        }

        };
        }
```

ManufacturerForm.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;
    using namespace System::IO;

    public ref class ManufacturerForm : public System::Windows::Forms::Form
    {
    public:
        ManufacturerForm(Form^ previousForm)
        {
            InitializeComponent();
            this->previousForm = previousForm;
        }

    protected:
        ~ManufacturerForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::TextBox^ txtName;
    private: System::Windows::Forms::TextBox^ txtCode;
```

```
        private:
System::Windows::Forms::DateTimePicker^
dtpManufactureDate;
        private:
System::Windows::Forms::DateTimePicker^
dtpExpiryDate;
        private:
System::Windows::Forms::ComboBox^ cmbStatus;
        private:
System::Windows::Forms::TextBox^
txtPharmacyName;
        private: System::Windows::Forms::Button^
btnRegisterMedicine;
        private: System::Windows::Forms::Button^
btnChooseFile;
        private:
System::Windows::Forms::TextBox^ txtFilePath;
        private: System::Windows::Forms::Button^
btnBack;
        private:
System::Windows::Forms::TextBox^ txtDescription;
        private: Form^ previousForm;
        private: System::Windows::Forms::Label^
label1;
        private: System::Windows::Forms::Label^
label2;
        private: System::Windows::Forms::Label^
label3;
        private: System::Windows::Forms::Label^
label4;
        private: System::Windows::Forms::Label^
label5;
        private: System::Windows::Forms::Label^
label6;



        private:
            System::ComponentModel::Container^
components;

        #pragma region Windows Form Designer
generated code
            void InitializeComponent(void)
            {
                this->txtName = (gcnew
System::Windows::Forms::TextBox());
                this->txtCode = (gcnew
System::Windows::Forms::TextBox());
                this->dtpManufactureDate = (gcnew
System::Windows::Forms::DateTimePicker());
                this->dtpExpiryDate = (gcnew
System::Windows::Forms::DateTimePicker());
                this->cmbStatus = (gcnew
System::Windows::Forms::ComboBox());
                this->txtPharmacyName = (gcnew
System::Windows::Forms::TextBox());
                this->btnRegisterMedicine = (gcnew
System::Windows::Forms::Button());
                this->btnChooseFile = (gcnew
System::Windows::Forms::Button());
                this->txtFilePath = (gcnew
System::Windows::Forms::TextBox());
                this->btnBack = (gcnew
System::Windows::Forms::Button());
                this->txtDescription = (gcnew
System::Windows::Forms::TextBox());
                this->label1 = (gcnew
System::Windows::Forms::Label());
                this->label2 = (gcnew
System::Windows::Forms::Label());
                this->label3 = (gcnew
System::Windows::Forms::Label());
                this->label4 = (gcnew
System::Windows::Forms::Label());
                this->label5 = (gcnew
System::Windows::Forms::Label());
                this->label6 = (gcnew
System::Windows::Forms::Label());
                this->SuspendLayout();
                //
                // txtName
                //
                this->txtName->Location =
System::Drawing::Point(30, 87);
                this->txtName->Name = L"txtName";
                this->txtName->Size =
System::Drawing::Size(200, 29);
                this->txtName->TabIndex = 0;
                //
                // txtCode
                //
                this->txtCode->Location =
System::Drawing::Point(30, 143);
                this->txtCode->Name = L"txtCode";
                this->txtCode->Size =
System::Drawing::Size(200, 29);
                this->txtCode->TabIndex = 1;
                //
                // dtpManufactureDate
                //
                this->dtpManufactureDate->Location =
System::Drawing::Point(30, 208);
                this->dtpManufactureDate->Name =
L"dtpManufactureDate";
                this->dtpManufactureDate->Size =
System::Drawing::Size(200, 29);
                this->dtpManufactureDate->TabIndex
= 2;
                //
                // dtpExpiryDate
                //
                this->dtpExpiryDate->Location =
System::Drawing::Point(30, 274);
                this->dtpExpiryDate->Name =
L"dtpExpiryDate";
```

```
            this->dtpExpiryDate->Size = System::Drawing::Size(200, 29);
            this->dtpExpiryDate->TabIndex = 3;
            //
            // cmbStatus
            //
            this->cmbStatus->FormattingEnabled = true;
            this->cmbStatus->Items->AddRange(gcnew cli::array< System::Object^ >(2) { L"У виробництві", L"Виготовлено" });
            this->cmbStatus->Location = System::Drawing::Point(30, 339);
            this->cmbStatus->Name = L"cmbStatus";
            this->cmbStatus->Size = System::Drawing::Size(200, 29);
            this->cmbStatus->TabIndex = 4;
            //
            // txtPharmacyName
            //
            this->txtPharmacyName->Location = System::Drawing::Point(30, 374);
            this->txtPharmacyName->Name = L"txtPharmacyName";
            this->txtPharmacyName->Size = System::Drawing::Size(200, 29);
            this->txtPharmacyName->TabIndex = 5;
            this->txtPharmacyName->Visible = false;
            //
            // btnRegisterMedicine
            //
            this->btnRegisterMedicine->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnRegisterMedicine->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->btnRegisterMedicine->ForeColor = System::Drawing::SystemColors::Control;
            this->btnRegisterMedicine->Location = System::Drawing::Point(30, 447);
            this->btnRegisterMedicine->Name = L"btnRegisterMedicine";
            this->btnRegisterMedicine->Size = System::Drawing::Size(133, 35);
            this->btnRegisterMedicine->TabIndex = 8;
            this->btnRegisterMedicine->Text = L"Реєстрація ліків";
            this->btnRegisterMedicine->UseVisualStyleBackColor = false;
            this->btnRegisterMedicine->Click += gcnew System::EventHandler(this, &ManufacturerForm::btnRegisterMedicine_Click);
            //
            // btnChooseFile
            //
            this->btnChooseFile->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnChooseFile->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->btnChooseFile->ForeColor = System::Drawing::SystemColors::Control;
            this->btnChooseFile->Location = System::Drawing::Point(383, 443);
            this->btnChooseFile->Name = L"btnChooseFile";
            this->btnChooseFile->Size = System::Drawing::Size(186, 35);
            this->btnChooseFile->TabIndex = 6;
            this->btnChooseFile->Text = L"Вибрати сертифікат";
            this->btnChooseFile->UseVisualStyleBackColor = false;
            this->btnChooseFile->Click += gcnew System::EventHandler(this, &ManufacturerForm::btnChooseFile_Click);
            //
            // txtFilePath
            //
            this->txtFilePath->Location = System::Drawing::Point(575, 443);
            this->txtFilePath->Multiline = true;
            this->txtFilePath->Name = L"txtFilePath";
            this->txtFilePath->ReadOnly = true;
            this->txtFilePath->Size = System::Drawing::Size(209, 33);
            this->txtFilePath->TabIndex = 7;
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(30, 12);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(133, 33);
            this->btnBack->TabIndex = 9;
            this->btnBack->Text = L"Повернутися";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &ManufacturerForm::btnBack_Click);
            //
            // txtDescription
            //
```

```
            this->txtDescription->Location = System::Drawing::Point(386, 35);
            this->txtDescription->Multiline = true;
            this->txtDescription->Name = L"txtDescription";
            this->txtDescription->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;
            this->txtDescription->Size = System::Drawing::Size(398, 368);
            this->txtDescription->TabIndex = 10;
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Location = System::Drawing::Point(26, 63);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(52, 21);
            this->label1->TabIndex = 11;
            this->label1->Text = L"Назва";
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Location = System::Drawing::Point(26, 119);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(37, 21);
            this->label2->TabIndex = 12;
            this->label2->Text = L"Код";
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Location = System::Drawing::Point(26, 184);
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(143, 21);
            this->label3->TabIndex = 13;
            this->label3->Text = L"Дата виробництва";
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Location = System::Drawing::Point(26, 250);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(150, 21);
            this->label4->TabIndex = 14;
            this->label4->Text = L"Термін придатності";
            //
            // label5
            //
            this->label5->AutoSize = true;
            this->label5->Location = System::Drawing::Point(26, 315);
            this->label5->Name = L"label5";
            this->label5->Size = System::Drawing::Size(57, 21);
            this->label5->TabIndex = 15;
            this->label5->Text = L"Статус";
            //
            // label6
            //
            this->label6->AutoSize = true;
            this->label6->Location = System::Drawing::Point(382, 11);
            this->label6->Name = L"label6";
            this->label6->Size = System::Drawing::Size(126, 21);
            this->label6->TabIndex = 16;
            this->label6->Text = L"Опис препарату";
            //
            // ManufacturerForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label6);
            this->Controls->Add(this->label5);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->txtDescription);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->btnRegisterMedicine);
            this->Controls->Add(this->txtFilePath);
            this->Controls->Add(this->btnChooseFile);
            this->Controls->Add(this->txtPharmacyName);
            this->Controls->Add(this->cmbStatus);
            this->Controls->Add(this->dtpExpiryDate);
            this->Controls->Add(this->dtpManufactureDate);
            this->Controls->Add(this->txtCode);
            this->Controls->Add(this->txtName);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
```

```
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"ManufacturerForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void btnChooseFile_Click(System::Object^ sender, System::EventArgs^ e) {
        OpenFileDialog^ openFileDialog = gcnew OpenFileDialog();
        openFileDialog->Filter = "PDF Files|*.pdf";
        openFileDialog->Title = "Вибрати PDF файл";

        if (openFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
            txtFilePath->Text = openFileDialog->FileName;
        }
    }

    private: System::Void btnRegisterMedicine_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ name = txtName->Text;
        String^ code = txtCode->Text;
        DateTime manufactureDate = dtpManufactureDate->Value;
        DateTime expiryDate = dtpExpiryDate->Value;
        String^ status = cmbStatus->SelectedItem->ToString();
        String^ description = txtDescription->Text;
        String^ filePath = txtFilePath->Text;

        if (String::IsNullOrEmpty(filePath)) {
            MessageBox::Show("Будь ласка, виберіть PDF файл");
            return;
        }

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("INSERT INTO Medicines (Name, Code, ManufactureDate, ExpiryDate, Status, Description, FilePath) VALUES (@name, @code, @manufactureDate, @expiryDate, @status, @description, @filePath)", conn);

        cmd->Parameters->AddWithValue("@name", name);
        cmd->Parameters->AddWithValue("@code", code);
        cmd->Parameters->AddWithValue("@manufactureDate", manufactureDate);
        cmd->Parameters->AddWithValue("@expiryDate", expiryDate);
        cmd->Parameters->AddWithValue("@status", status);
        cmd->Parameters->AddWithValue("@description", description);
        cmd->Parameters->AddWithValue("@filePath", filePath);

        try {
            conn->Open();
            cmd->ExecuteNonQuery();
            MessageBox::Show("Ліки зареєстровані успішно");
        }
        catch (Exception^ ex) {
            MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
        }
        finally {
            conn->Close();
        }
    }

    private: System::Void btnBack_Click(System::Object^ sender, System::EventArgs^ e) {
        this->Close();
        previousForm->Show();
    }
    private: System::Void richTextBox1_TextChanged(System::Object^ sender, System::EventArgs^ e) {
    }
};
}
```

ManufacturerMenuForm.h

```
#pragma once
#include "ManufacturerForm.h"
#include "TrackMedicineForm.h"
```

```
#include "ReportForm.h"

namespace MedicationProtectionApp {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    public ref class ManufacturerMenuForm : public System::Windows::Forms::Form
    {
    public:
        ManufacturerMenuForm(String^ username, Form^ loginForm)
        {
            InitializeComponent();
            previousForm = loginForm;
            lblWelcome->Text = "Вітаю, " + username + ", ви авторизувались як виробник";
            CenterLabel(); // Центруємо текст після ініціалізації
        }

    protected:
        ~ManufacturerMenuForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^ lblWelcome;
    private: System::Windows::Forms::Button^ btnRegisterMedicine;
    private: System::Windows::Forms::Button^ btnTrackMedicine;
    private: System::Windows::Forms::Button^ btnGenerateReport;
    private: System::Windows::Forms::TextBox^ txtReportCode;
    private: System::Windows::Forms::Button^ btnBack;

    private:
        Form^ previousForm;
    private: System::Windows::Forms::Label^ label1;

        System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
            this->lblWelcome = (gcnew System::Windows::Forms::Label());
            this->btnRegisterMedicine = (gcnew System::Windows::Forms::Button());
            this->btnTrackMedicine = (gcnew System::Windows::Forms::Button());
            this->btnGenerateReport = (gcnew System::Windows::Forms::Button());
            this->txtReportCode = (gcnew System::Windows::Forms::TextBox());
            this->btnBack = (gcnew System::Windows::Forms::Button());
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // lblWelcome
            //
            this->lblWelcome->AutoSize = true;
            this->lblWelcome->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Bold));
            this->lblWelcome->Location = System::Drawing::Point(0, 0);
            this->lblWelcome->Name = L"lblWelcome";
            this->lblWelcome->Size = System::Drawing::Size(0, 21);
            this->lblWelcome->TabIndex = 0;
            //
            // btnRegisterMedicine
            //
            this->btnRegisterMedicine->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnRegisterMedicine->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnRegisterMedicine->ForeColor = System::Drawing::SystemColors::Control;
            this->btnRegisterMedicine->Location = System::Drawing::Point(297, 226);
            this->btnRegisterMedicine->Name = L"btnRegisterMedicine";
            this->btnRegisterMedicine->Size = System::Drawing::Size(212, 53);
            this->btnRegisterMedicine->TabIndex = 1;
            this->btnRegisterMedicine->Text = L"Зареєструвати ліки";
            this->btnRegisterMedicine->UseVisualStyleBackColor = false;
```

```
            this->btnRegisterMedicine->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnRegisterMedicine_Click);
            //
            // btnTrackMedicine
            //
            this->btnTrackMedicine->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnTrackMedicine->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnTrackMedicine->ForeColor = System::Drawing::SystemColors::Control;
            this->btnTrackMedicine->Location = System::Drawing::Point(297, 301);
            this->btnTrackMedicine->Name = L"btnTrackMedicine";
            this->btnTrackMedicine->Size = System::Drawing::Size(214, 53);
            this->btnTrackMedicine->TabIndex = 2;
            this->btnTrackMedicine->Text = L"Відстежити ліки";
            this->btnTrackMedicine->UseVisualStyleBackColor = false;
            this->btnTrackMedicine->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnTrackMedicine_Click);
            //
            // btnGenerateReport
            //
            this->btnGenerateReport->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnGenerateReport->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnGenerateReport->ForeColor = System::Drawing::SystemColors::Control;
            this->btnGenerateReport->Location = System::Drawing::Point(326, 428);
            this->btnGenerateReport->Name = L"btnGenerateReport";
            this->btnGenerateReport->Size = System::Drawing::Size(160, 57);
            this->btnGenerateReport->TabIndex = 4;
            this->btnGenerateReport->Text = L"Згенерувати звіт";
            this->btnGenerateReport->UseVisualStyleBackColor = false;
            this->btnGenerateReport->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnGenerateReport_Click);
            //
            // txtReportCode
            //
            this->txtReportCode->Location = System::Drawing::Point(297, 377);
            this->txtReportCode->Name = L"txtReportCode";
            this->txtReportCode->Size = System::Drawing::Size(214, 29);
            this->txtReportCode->TabIndex = 3;
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(297, 149);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(212, 53);
            this->btnBack->TabIndex = 5;
            this->btnBack->Text = L"<-- Повернутися назад";
            this->btnBack->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
            this->label1->Location = System::Drawing::Point(309, 78);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(188, 45);
            this->label1->TabIndex = 6;
            this->label1->Text = L"DrugGuard";
            //
            // ManufacturerMenuForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->btnGenerateReport);
            this->Controls->Add(this->txtReportCode);
```

```
            this->Controls->Add(this->btnTrackMedicine);
            this->Controls->Add(this->btnRegisterMedicine);
            this->Controls->Add(this->lblWelcome);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"ManufacturerMenuForm";
            this->Text = L"DrugGuard - виробник";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

        private: void CenterLabel()
        {
            lblWelcome->Location = System::Drawing::Point((this->ClientSize.Width - lblWelcome->Width) / 2, lblWelcome->Location.Y);
        }

        private: System::Void btnRegisterMedicine_Click(System::Object^ sender, System::EventArgs^ e) {
            ManufacturerForm^ mf = gcnew ManufacturerForm(this);
            mf->ShowDialog();
        }

        private: System::Void btnTrackMedicine_Click(System::Object^ sender, System::EventArgs^ e) {
            TrackMedicineForm^ tmf = gcnew TrackMedicineForm(this);
            tmf->ShowDialog();
        }

        private: System::Void btnGenerateReport_Click(System::Object^ sender, System::EventArgs^ e) {
            String^ code = txtReportCode->Text;

            SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
            SqlCommand^ cmd = gcnew SqlCommand("SELECT Name, Code, ManufactureDate, ExpiryDate, Status, Description, FilePath FROM Medicines WHERE Code=@code", conn);

            cmd->Parameters->AddWithValue("@code", code);

            SqlDataReader^ reader;

            try {
                conn->Open();
                reader = cmd->ExecuteReader();

                if (reader->Read()) {
                    String^ name = reader["Name"]->ToString();
                    String^ status = reader["Status"]->ToString();
                    String^ manufactureDate = reader["ManufactureDate"]->ToString();
                    String^ expiryDate = reader["ExpiryDate"]->ToString();
                    String^ description = reader["Description"]->ToString();
                    String^ filePath = reader["FilePath"]->ToString();

                    String^ report = "Звіт по препарату \"" + name + "\" успішно згенеровано!\n\n" +
                        "Назва препарату: " + name + "\n" +
                        "Код препарату: " + code + "\n" +
                        "Дата виробництва: " + manufactureDate + "\n" +
                        "Термін придатності: " + expiryDate + "\n" +
                        "Статус: " + status + "\n" +
                        "Опис: " + description;

                    ReportForm^ reportForm = gcnew ReportForm(report, filePath);
                    reportForm->ShowDialog();
                }
                else {
                    MessageBox::Show("Ліки не знайдено.");
                }
            }
            catch (Exception^ ex) {
                MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
            }
            finally {
                if (reader != nullptr) {
                    reader->Close();
                }
                conn->Close();
            }
```

```
        }

        private: System::Void
btnBack_Click(System::Object^ sender,
System::EventArgs^ e) {
            this->Close();
            previousForm->Show();
        }
        };
}
```

PharmacyMenuForm.h

```
#pragma once

#include "ManufacturerForm.h"
#include "TrackMedicineForm.h"
#include "ReportForm.h"

namespace MedicationProtectionApp {

    using namespace System;
    using namespace
System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    public ref class ManufacturerMenuForm :
public System::Windows::Forms::Form
    {
    public:
        ManufacturerMenuForm(String^
username, Form^ loginForm)
        {
            InitializeComponent();
            previousForm = loginForm;
            lblWelcome->Text = "Вітаю, " +
username + ", ви авторизувались як виробник";
            CenterLabel(); // Центруємо текст
після ініціалізації
        }

    protected:
        ~ManufacturerMenuForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Label^
lblWelcome;
    private: System::Windows::Forms::Button^
btnRegisterMedicine;
    private: System::Windows::Forms::Button^
btnTrackMedicine;
    private: System::Windows::Forms::Button^
btnGenerateReport;
    private:
System::Windows::Forms::TextBox^ txtReportCode;
    private: System::Windows::Forms::Button^
btnBack;

    private:
        Form^ previousForm;
    private: System::Windows::Forms::Label^
label1;

        System::ComponentModel::Container^
components;

    #pragma region Windows Form Designer
generated code
        void InitializeComponent(void)
        {
            this->lblWelcome = (gcnew
System::Windows::Forms::Label());
            this->btnRegisterMedicine = (gcnew
System::Windows::Forms::Button());
            this->btnTrackMedicine = (gcnew
System::Windows::Forms::Button());
            this->btnGenerateReport = (gcnew
System::Windows::Forms::Button());
            this->txtReportCode = (gcnew
System::Windows::Forms::TextBox());
            this->btnBack = (gcnew
System::Windows::Forms::Button());
            this->label1 = (gcnew
System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // lblWelcome
            //
            this->lblWelcome->AutoSize = true;
            this->lblWelcome->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12,
System::Drawing::FontStyle::Bold));
            this->lblWelcome->Location =
System::Drawing::Point(0, 0);
            this->lblWelcome->Name =
L"lblWelcome";
            this->lblWelcome->Size =
System::Drawing::Size(0, 21);
            this->lblWelcome->TabIndex = 0;
            //
            // btnRegisterMedicine
            //
```

```
this->btnRegisterMedicine->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
this->btnRegisterMedicine->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
this->btnRegisterMedicine->ForeColor = System::Drawing::SystemColors::Control;
this->btnRegisterMedicine->Location = System::Drawing::Point(297, 226);
this->btnRegisterMedicine->Name = L"btnRegisterMedicine";
this->btnRegisterMedicine->Size = System::Drawing::Size(212, 53);
this->btnRegisterMedicine->TabIndex = 1;
this->btnRegisterMedicine->Text = L"Зареєструвати ліки";
this->btnRegisterMedicine->UseVisualStyleBackColor = false;
this->btnRegisterMedicine->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnRegisterMedicine_Click);
//
// btnTrackMedicine
//
this->btnTrackMedicine->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
this->btnTrackMedicine->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
this->btnTrackMedicine->ForeColor = System::Drawing::SystemColors::Control;
this->btnTrackMedicine->Location = System::Drawing::Point(297, 301);
this->btnTrackMedicine->Name = L"btnTrackMedicine";
this->btnTrackMedicine->Size = System::Drawing::Size(214, 53);
this->btnTrackMedicine->TabIndex = 2;
this->btnTrackMedicine->Text = L"Відстежити ліки";
this->btnTrackMedicine->UseVisualStyleBackColor = false;
this->btnTrackMedicine->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnTrackMedicine_Click);
//
// btnGenerateReport
//
this->btnGenerateReport->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
this->btnGenerateReport->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
this->btnGenerateReport->ForeColor = System::Drawing::SystemColors::Control;
this->btnGenerateReport->Location = System::Drawing::Point(326, 428);
this->btnGenerateReport->Name = L"btnGenerateReport";
this->btnGenerateReport->Size = System::Drawing::Size(160, 57);
this->btnGenerateReport->TabIndex = 4;
this->btnGenerateReport->Text = L"Згенерувати звіт";
this->btnGenerateReport->UseVisualStyleBackColor = false;
this->btnGenerateReport->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnGenerateReport_Click);
//
// txtReportCode
//
this->txtReportCode->Location = System::Drawing::Point(297, 377);
this->txtReportCode->Name = L"txtReportCode";
this->txtReportCode->Size = System::Drawing::Size(214, 29);
this->txtReportCode->TabIndex = 3;
//
// btnBack
//
this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
this->btnBack->Location = System::Drawing::Point(297, 149);
this->btnBack->Name = L"btnBack";
this->btnBack->Size = System::Drawing::Size(212, 53);
this->btnBack->TabIndex = 5;
this->btnBack->Text = L"<-- Повернутися назад";
this->btnBack->TextAlign = System::Drawing::ContentAlignment::MiddleLeft;
this->btnBack->UseVisualStyleBackColor = false;
this->btnBack->Click += gcnew System::EventHandler(this, &ManufacturerMenuForm::btnBack_Click);
//
// label1
//
this->label1->AutoSize = true;
this->label1->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
```

```
            this->label1->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 24,
System::Drawing::FontStyle::Bold));
            this->label1->Location =
System::Drawing::Point(309, 78);
            this->label1->Name = L"label1";
            this->label1->Size =
System::Drawing::Size(188, 45);
            this->label1->TabIndex = 6;
            this->label1->Text = L"DrugGuard";
            //
            // ManufacturerMenuForm
            //
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize =
System::Drawing::Size(796, 521);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this-
>btnGenerateReport);
            this->Controls->Add(this-
>txtReportCode);
            this->Controls->Add(this-
>btnTrackMedicine);
            this->Controls->Add(this-
>btnRegisterMedicine);
            this->Controls->Add(this-
>lblWelcome);
            this->Font = (gcnew
System::Drawing::Font(L"Segoe UI", 12,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedToo
lWindow;
            this->Name =
L"ManufacturerMenuForm";
            this->Text = L"DrugGuard -
виробник";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

        private: void CenterLabel()
        {
            lblWelcome->Location =
System::Drawing::Point((this->ClientSize.Width -
lblWelcome->Width) / 2, lblWelcome->Location.Y);
        }

        private: System::Void
btnRegisterMedicine_Click(System::Object^ sender,
System::EventArgs^ e) {
            ManufacturerForm^ mf = gcnew
ManufacturerForm(this);
            mf->ShowDialog();
        }

        private: System::Void
btnTrackMedicine_Click(System::Object^ sender,
System::EventArgs^ e) {
            TrackMedicineForm^ tmf = gcnew
TrackMedicineForm(this);
            tmf->ShowDialog();
        }

        private: System::Void
btnGenerateReport_Click(System::Object^ sender,
System::EventArgs^ e) {
            String^ code = txtReportCode->Text;

            SqlConnection^ conn = gcnew
SqlConnection("Server=DESKTOP-
ACME1K0;Database=MedicationSystem;Integrated
Security=True;");
            SqlCommand^ cmd = gcnew
SqlCommand("SELECT Name, Code,
ManufactureDate, ExpiryDate, Status, Description,
FilePath FROM Medicines WHERE Code=@code",
conn);

            cmd->Parameters-
>AddWithValue("@code", code);

            SqlDataReader^ reader;

            try {
                conn->Open();
                reader = cmd->ExecuteReader();

                if (reader->Read()) {
                    String^ name = reader["Name"]-
>ToString();
                    String^ status = reader["Status"]-
>ToString();
                    String^ manufactureDate =
reader["ManufactureDate"]->ToString();
                    String^ expiryDate =
reader["ExpiryDate"]->ToString();
                    String^ description =
reader["Description"]->ToString();
                    String^ filePath = reader["FilePath"]-
>ToString();

                    String^ report = "Звіт по препарату
\"" + name + "\" успішно згенеровано!\n\n" +
                        "Назва препарату: " + name + "\n"
+
                        "Код препарату: " + code + "\n" +
                        "Дата виробництва: " +
manufactureDate + "\n" +
```

```
                    "Термін придатності: " +
expiryDate + "\n" +
                    "Статус: " + status + "\n" +
                    "Опис: " + description;

                ReportForm^ reportForm = gcnew
ReportForm(report, filePath);
                reportForm->ShowDialog();
            }
            else {
                MessageBox::Show("Ліки не
знайдено.");
            }
        }
        catch (Exception^ ex) {
            MessageBox::Show("Помилка
підключення до бази даних: " + ex->Message);
        }
        finally {
            if (reader != nullptr) {
                reader->Close();
            }
            conn->Close();
        }
    }

    private: System::Void
btnBack_Click(System::Object^ sender,
System::EventArgs^ e) {
        this->Close();
        previousForm->Show();
    }
    };
}
```

RegisterForm.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;
    using namespace
System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;

    public ref class RegisterForm : public
System::Windows::Forms::Form
    {
    public:
        RegisterForm(void)
        {
            InitializeComponent();
        }

    protected:
        ~RegisterForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private:
System::Windows::Forms::TextBox^ txtLogin;
    private:
System::Windows::Forms::TextBox^ txtPassword;
    private:
System::Windows::Forms::ComboBox^ cmbRole;
    private: System::Windows::Forms::Button^
btnRegister;
    private: System::Windows::Forms::Label^
label1;
    private: System::Windows::Forms::Label^
label2;
    private: System::Windows::Forms::Label^
label3;
    private: System::Windows::Forms::Label^
label4;

    private:
        System::ComponentModel::Container^
components;

    #pragma region Windows Form Designer
generated code
        void InitializeComponent(void)
        {
            this->txtLogin = (gcnew
System::Windows::Forms::TextBox());
            this->txtPassword = (gcnew
System::Windows::Forms::TextBox());
            this->cmbRole = (gcnew
System::Windows::Forms::ComboBox());
            this->btnRegister = (gcnew
System::Windows::Forms::Button());
            this->label1 = (gcnew
System::Windows::Forms::Label());
            this->label2 = (gcnew
System::Windows::Forms::Label());
            this->label3 = (gcnew
System::Windows::Forms::Label());
            this->label4 = (gcnew
System::Windows::Forms::Label());
            this->SuspendLayout();
            //
```

```
            // txtLogin
            //
            this->txtLogin->Location = System::Drawing::Point(269, 190);
            this->txtLogin->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->txtLogin->Name = L"txtLogin";
            this->txtLogin->Size = System::Drawing::Size(298, 29);
            this->txtLogin->TabIndex = 0;
            //
            // txtPassword
            //
            this->txtPassword->Location = System::Drawing::Point(269, 276);
            this->txtPassword->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->txtPassword->Name = L"txtPassword";
            this->txtPassword->PasswordChar = '*';
            this->txtPassword->Size = System::Drawing::Size(298, 29);
            this->txtPassword->TabIndex = 1;
            this->txtPassword->UseSystemPasswordChar = true;
            //
            // cmbRole
            //
            this->cmbRole->FormattingEnabled = true;
            this->cmbRole->Items->AddRange(gcnew cli::array< System::Object^ >(3) { L"Виробник", L"Аптека", L"Споживач" });
            this->cmbRole->Location = System::Drawing::Point(269, 356);
            this->cmbRole->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->cmbRole->Name = L"cmbRole";
            this->cmbRole->Size = System::Drawing::Size(298, 29);
            this->cmbRole->TabIndex = 2;
            //
            // btnRegister
            //
            this->btnRegister->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnRegister->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnRegister->ForeColor = System::Drawing::SystemColors::Control;
            this->btnRegister->Location = System::Drawing::Point(326, 435);
            this->btnRegister->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->btnRegister->Name = L"btnRegister";
            this->btnRegister->Size = System::Drawing::Size(146, 54);
            this->btnRegister->TabIndex = 3;
            this->btnRegister->Text = L"Реєстрація";
            this->btnRegister->UseVisualStyleBackColor = false;
            this->btnRegister->Click += gcnew System::EventHandler(this, &RegisterForm::btnRegister_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Location = System::Drawing::Point(265, 164);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(103, 21);
            this->label1->TabIndex = 4;
            this->label1->Text = L"Введіть логін";
            this->label1->Click += gcnew System::EventHandler(this, &RegisterForm::label1_Click);
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Location = System::Drawing::Point(265, 250);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(118, 21);
            this->label2->TabIndex = 5;
            this->label2->Text = L"Введіть пароль";
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Location = System::Drawing::Point(265, 330);
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(114, 21);
            this->label3->TabIndex = 6;
            this->label3->Text = L"Вкажіть хтови";
            this->label3->Click += gcnew System::EventHandler(this, &RegisterForm::label3_Click);
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
```

```
            this->label4->Location = System::Drawing::Point(318, 102);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(188, 45);
            this->label4->TabIndex = 7;
            this->label4->Text = L"DrugGuard";
            //
            // RegisterForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnRegister);
            this->Controls->Add(this->cmbRole);
            this->Controls->Add(this->txtPassword);
            this->Controls->Add(this->txtLogin);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Margin = System::Windows::Forms::Padding(4, 5, 4, 5);
            this->Name = L"RegisterForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void btnRegister_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ login = txtLogin->Text;
        String^ password = txtPassword->Text;
        String^ role = cmbRole->SelectedItem->ToString();

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("INSERT INTO Users (Login, Password, Role) VALUES (@login, @password, @role)", conn);

        cmd->Parameters->AddWithValue("@login", login);
        cmd->Parameters->AddWithValue("@password", password);
        cmd->Parameters->AddWithValue("@role", role);

        try {
            conn->Open();
            cmd->ExecuteNonQuery();
            MessageBox::Show("Реєстрація успішна");
            this->Close(); // Після успішної реєстрації повертаємося до форми входу
        }
        catch (Exception^ ex) {
            MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
        }
        finally {
            conn->Close();
        }
    }

    private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
    }
    private: System::Void label3_Click(System::Object^ sender, System::EventArgs^ e) {
    }
};
}
```

ReportForm.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::IO;
```

```
public ref class ReportForm : public System::Windows::Forms::Form
{
public:
    ReportForm(String^ report, String^ certificatePath)
    {
        InitializeComponent();
        lnkCertificate->Text = "Сертифікат";
        lnkCertificate->Tag = certificatePath;
        txtReport->SelectionStart = 0;
        txtReport->SelectionLength = 0;
        txtReport->ScrollToCaret();

        String^ reportTitle = ExtractReportTitle(report);
        lblReportTitle->Text = reportTitle;
        CenterLabel(lblReportTitle); // Центруємо текст після ініціалізації
        txtReport->Text = report->Substring(reportTitle->Length + 2); // +2 to remove the '\n\n' after the title
    }

protected:
    ~ReportForm()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::RichTextBox^ txtReport;
private: System::Windows::Forms::LinkLabel^ lnkCertificate;
private: System::Windows::Forms::Label^ lblReportTitle;

private:
    System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
    void InitializeComponent(void)
    {
        this->txtReport = (gcnew System::Windows::Forms::RichTextBox());
        this->lnkCertificate = (gcnew System::Windows::Forms::LinkLabel());
        this->lblReportTitle = (gcnew System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // txtReport
        //
        this->txtReport->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12));
        this->txtReport->Location = System::Drawing::Point(20, 60);
        this->txtReport->Name = L"txtReport";
        this->txtReport->ReadOnly = true;
        this->txtReport->ScrollBars = System::Windows::Forms::RichTextBoxScrollBars::Vertical;
        this->txtReport->Size = System::Drawing::Size(760, 420);
        this->txtReport->TabIndex = 0;
        this->txtReport->Text = L"";
        //
        // lnkCertificate
        //
        this->lnkCertificate->AutoSize = true;
        this->lnkCertificate->Location = System::Drawing::Point(20, 500);
        this->lnkCertificate->Name = L"lnkCertificate";
        this->lnkCertificate->Size = System::Drawing::Size(0, 21);
        this->lnkCertificate->TabIndex = 1;
        this->lnkCertificate->LinkClicked += gcnew System::Windows::Forms::LinkLabelLinkClickedEventHandler(this, &ReportForm::lnkCertificate_LinkClicked);
        //
        // lblReportTitle
        //
        this->lblReportTitle->AutoSize = true;
        this->lblReportTitle->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Bold));
        this->lblReportTitle->Location = System::Drawing::Point(0, 20);
        this->lblReportTitle->Name = L"lblReportTitle";
        this->lblReportTitle->Size = System::Drawing::Size(0, 21);
        this->lblReportTitle->TabIndex = 2;
        //
        // ReportForm
        //
        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
        this->ClientSize = System::Drawing::Size(812, 560);
        this->Controls->Add(this->lblReportTitle);
        this->Controls->Add(this->lnkCertificate);
        this->Controls->Add(this->txtReport);
        this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12));
```

```
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"ReportForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

        private: System::Void lnkCertificate_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {
            String^ filePath = lnkCertificate->Tag->ToString();
            if (!String::IsNullOrEmpty(filePath) && System::IO::File::Exists(filePath)) {
                System::Diagnostics::Process::Start(filePath);
            }
            else {
                MessageBox::Show("Файл не знайдено.");
            }
        }

        private: String^ ExtractReportTitle(String^ report)
        {
            array<String^>^ lines = report->Split('\n');
            return lines[0];
        }

        private: void CenterLabel(System::Windows::Forms::Label^ label)
        {
            label->Location = System::Drawing::Point((this->ClientSize.Width - label->Width) / 2, label->Location.Y);
        }
        };
    }
```

TrackMedicineForm.h

```
    #pragma once

    namespace MedicationProtectionApp {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
        using namespace System::Data::SqlClient;
        using namespace System::IO;

        public ref class TrackMedicineForm : public System::Windows::Forms::Form
        {
        public:
            TrackMedicineForm(Form^ previousForm)
            {
                InitializeComponent();
                this->previousForm = previousForm;
            }

        protected:
            ~TrackMedicineForm()
            {
                if (components)
                {
                    delete components;
                }
            }

        private: System::Windows::Forms::TextBox^ txtCode;
        private: System::Windows::Forms::Button^ btnTrack;
        private: System::Windows::Forms::Label^ lblName;
        private: System::Windows::Forms::Label^ lblCode;
        private: System::Windows::Forms::Label^ lblManufactureDate;
        private: System::Windows::Forms::Label^ lblExpiryDate;
        private: System::Windows::Forms::Label^ lblStatus;
        private: System::Windows::Forms::LinkLabel^ lnkFile;
        private: System::Windows::Forms::Button^ btnBack;
        private: Form^ previousForm;
        private: System::Windows::Forms::Label^ label1;
        private: System::Windows::Forms::Label^ label2;

        private:
            System::ComponentModel::Container^ components;
```

```
        #pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
            this->txtCode = (gcnew System::Windows::Forms::TextBox());
            this->btnTrack = (gcnew System::Windows::Forms::Button());
            this->lblName = (gcnew System::Windows::Forms::Label());
            this->lblCode = (gcnew System::Windows::Forms::Label());
            this->lblManufactureDate = (gcnew System::Windows::Forms::Label());
            this->lblExpiryDate = (gcnew System::Windows::Forms::Label());
            this->lblStatus = (gcnew System::Windows::Forms::Label());
            this->lnkFile = (gcnew System::Windows::Forms::LinkLabel());
            this->btnBack = (gcnew System::Windows::Forms::Button());
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // txtCode
            //
            this->txtCode->Location = System::Drawing::Point(170, 209);
            this->txtCode->Name = L"txtCode";
            this->txtCode->Size = System::Drawing::Size(484, 29);
            this->txtCode->TabIndex = 0;
            //
            // btnTrack
            //
            this->btnTrack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnTrack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnTrack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnTrack->Location = System::Drawing::Point(353, 436);
            this->btnTrack->Name = L"btnTrack";
            this->btnTrack->Size = System::Drawing::Size(123, 54);
            this->btnTrack->TabIndex = 1;
            this->btnTrack->Text = L"Відстежити";
            this->btnTrack->UseVisualStyleBackColor = false;
            this->btnTrack->Click += gcnew System::EventHandler(this, &TrackMedicineForm::btnTrack_Click);
            //
            // lblName
            //
            this->lblName->AutoSize = true;
            this->lblName->Location = System::Drawing::Point(170, 250);
            this->lblName->Name = L"lblName";
            this->lblName->Size = System::Drawing::Size(0, 21);
            this->lblName->TabIndex = 2;
            //
            // lblCode
            //
            this->lblCode->AutoSize = true;
            this->lblCode->Location = System::Drawing::Point(170, 280);
            this->lblCode->Name = L"lblCode";
            this->lblCode->Size = System::Drawing::Size(0, 21);
            this->lblCode->TabIndex = 3;
            //
            // lblManufactureDate
            //
            this->lblManufactureDate->AutoSize = true;
            this->lblManufactureDate->Location = System::Drawing::Point(170, 310);
            this->lblManufactureDate->Name = L"lblManufactureDate";
            this->lblManufactureDate->Size = System::Drawing::Size(0, 21);
            this->lblManufactureDate->TabIndex = 4;
            //
            // lblExpiryDate
            //
            this->lblExpiryDate->AutoSize = true;
            this->lblExpiryDate->Location = System::Drawing::Point(170, 340);
            this->lblExpiryDate->Name = L"lblExpiryDate";
            this->lblExpiryDate->Size = System::Drawing::Size(0, 21);
            this->lblExpiryDate->TabIndex = 5;
            //
            // lblStatus
            //
            this->lblStatus->AutoSize = true;
            this->lblStatus->Location = System::Drawing::Point(170, 370);
            this->lblStatus->Name = L"lblStatus";
            this->lblStatus->Size = System::Drawing::Size(0, 21);
            this->lblStatus->TabIndex = 6;
            //
            // lnkFile
            //
            this->lnkFile->AutoSize = true;
```

```
            this->lnkFile->Location = System::Drawing::Point(170, 400);
            this->lnkFile->Name = L"lnkFile";
            this->lnkFile->Size = System::Drawing::Size(0, 21);
            this->lnkFile->TabIndex = 7;
            this->lnkFile->LinkClicked += gcnew System::Windows::Forms::LinkLabelLinkClickedEventHandler(this, &TrackMedicineForm::lnkFile_LinkClicked);
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(12, 12);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(184, 48);
            this->btnBack->TabIndex = 8;
            this->btnBack->Text = L"<-- Повернутися назад";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &TrackMedicineForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label1->Location = System::Drawing::Point(258, 175);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(326, 21);
            this->label1->TabIndex = 9;
            this->label1->Text = L"Введіть код препарату для відстеження";
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label2->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
            this->label2->Location = System::Drawing::Point(317, 101);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(188, 45);
            this->label2->TabIndex = 10;
            this->label2->Text = L"DrugGuard";
            //
            // TrackMedicineForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->lnkFile);
            this->Controls->Add(this->lblStatus);
            this->Controls->Add(this->lblExpiryDate);
            this->Controls->Add(this->lblManufactureDate);
            this->Controls->Add(this->lblCode);
            this->Controls->Add(this->lblName);
            this->Controls->Add(this->btnTrack);
            this->Controls->Add(this->txtCode);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"TrackMedicineForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void btnTrack_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ code = txtCode->Text;

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT Name, Code, ManufactureDate, ExpiryDate, Status, FilePath FROM Medicines WHERE Code=@code", conn);
```

```
            cmd->Parameters->AddWithValue("@code", code);

            SqlDataReader^ reader;

            try {
                conn->Open();
                reader = cmd->ExecuteReader();

                if (reader->Read()) {
                    String^ name = reader["Name"]->ToString();
                    String^ status = reader["Status"]->ToString();
                    String^ manufactureDate = reader["ManufactureDate"]->ToString();
                    String^ expiryDate = reader["ExpiryDate"]->ToString();
                    String^ filePath = reader["FilePath"]->ToString();

                    lblName->Text = "Назва ліків: " + name;
                    lblCode->Text = "Код ліків: " + code;
                    lblManufactureDate->Text = "Дата виготовлення: " + manufactureDate;
                    lblExpiryDate->Text = "Термін придатності: " + expiryDate;
                    lblStatus->Text = "Статус: " + status;
                    lnkFile->Text = "Сертифікат: " + Path::GetFileName(filePath);
                    lnkFile->Tag = filePath;
                }
                else {
                    lblName->Text = "Ліки не знайдено.";
                    lblCode->Text = "";
                    lblManufactureDate->Text = "";
                    lblExpiryDate->Text = "";
                    lblStatus->Text = "";
                    lnkFile->Text = "";
                    lnkFile->Tag = nullptr;
                }
            }
            catch (Exception^ ex) {
                lblName->Text = "Помилка підключення до бази даних: " + ex->Message;
                lblCode->Text = "";
                lblManufactureDate->Text = "";
                lblExpiryDate->Text = "";
                lblStatus->Text = "";
                lnkFile->Text = "";
                lnkFile->Tag = nullptr;
            }
            finally {
                if (reader != nullptr) {
                    reader->Close();
                }
                conn->Close();
            }
        }

        private: System::Void lnkFile_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {
            String^ filePath = lnkFile->Tag->ToString();
            if (!String::IsNullOrEmpty(filePath) && File::Exists(filePath)) {
                System::Diagnostics::Process::Start(filePath);
            }
            else {
                MessageBox::Show("Файл не знайдено.");
            }
        }

        private: System::Void btnBack_Click(System::Object^ sender, System::EventArgs^ e) {
            this->Close();
            previousForm->Show();
        }

        private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
        }
        };
    }
```

UpdateMedicineStatusForm.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;
    using namespace System::IO;
```

```
public ref class TrackMedicineForm : public
System::Windows::Forms::Form
{
public:
   TrackMedicineForm(Form^
previousForm)
   {
      InitializeComponent();
      this->previousForm = previousForm;
   }

protected:
   ~TrackMedicineForm()
   {
      if (components)
      {
         delete components;
      }
   }

private:
System::Windows::Forms::TextBox^ txtCode;
private:   System::Windows::Forms::Button^
btnTrack;
private:    System::Windows::Forms::Label^
lblName;
private:    System::Windows::Forms::Label^
lblCode;
private:    System::Windows::Forms::Label^
lblManufactureDate;
private:    System::Windows::Forms::Label^
lblExpiryDate;
private:    System::Windows::Forms::Label^
lblStatus;
private:
System::Windows::Forms::LinkLabel^ lnkFile;
private:   System::Windows::Forms::Button^
btnBack;
private: Form^ previousForm;
private:    System::Windows::Forms::Label^
label1;
private:    System::Windows::Forms::Label^
label2;

private:
   System::ComponentModel::Container^
components;

#pragma region Windows Form Designer
generated code
   void InitializeComponent(void)
   {
      this->txtCode          =          (gcnew
System::Windows::Forms::TextBox());
      this->btnTrack         =          (gcnew
System::Windows::Forms::Button());
      this->lblName          =          (gcnew
System::Windows::Forms::Label());
      this->lblCode          =          (gcnew
System::Windows::Forms::Label());
      this->lblManufactureDate   =   (gcnew
System::Windows::Forms::Label());
      this->lblExpiryDate        =        (gcnew
System::Windows::Forms::Label());
      this->lblStatus        =          (gcnew
System::Windows::Forms::Label());
      this->lnkFile          =          (gcnew
System::Windows::Forms::LinkLabel());
      this->btnBack          =          (gcnew
System::Windows::Forms::Button());
      this->label1           =          (gcnew
System::Windows::Forms::Label());
      this->label2           =          (gcnew
System::Windows::Forms::Label());
      this->SuspendLayout();
      //
      // txtCode
      //
      this->txtCode->Location                =
System::Drawing::Point(170, 209);
      this->txtCode->Name = L"txtCode";
      this->txtCode->Size                    =
System::Drawing::Size(484, 29);
      this->txtCode->TabIndex = 0;
      //
      // btnTrack
      //
      this->btnTrack->BackColor              =
System::Drawing::SystemColors::ActiveCaptionText;
      this->btnTrack->FlatStyle              =
System::Windows::Forms::FlatStyle::Popup;
      this->btnTrack->ForeColor              =
System::Drawing::SystemColors::Control;
      this->btnTrack->Location               =
System::Drawing::Point(353, 436);
      this->btnTrack->Name = L"btnTrack";
      this->btnTrack->Size                   =
System::Drawing::Size(123, 54);
      this->btnTrack->TabIndex = 1;
      this->btnTrack->Text                   =
L"Відстежити";
      this->btnTrack-
>UseVisualStyleBackColor = false;
      this->btnTrack->Click      +=      gcnew
System::EventHandler(this,
&TrackMedicineForm::btnTrack_Click);
      //
      // lblName
      //
      this->lblName->AutoSize = true;
      this->lblName->Location                =
System::Drawing::Point(170, 250);
      this->lblName->Name = L"lblName";
      this->lblName->Size                    =
System::Drawing::Size(0, 21);
      this->lblName->TabIndex = 2;
```

```
            //
            // lblCode
            //
            this->lblCode->AutoSize = true;
            this->lblCode->Location = System::Drawing::Point(170, 280);
            this->lblCode->Name = L"lblCode";
            this->lblCode->Size = System::Drawing::Size(0, 21);
            this->lblCode->TabIndex = 3;
            //
            // lblManufactureDate
            //
            this->lblManufactureDate->AutoSize = true;
            this->lblManufactureDate->Location = System::Drawing::Point(170, 310);
            this->lblManufactureDate->Name = L"lblManufactureDate";
            this->lblManufactureDate->Size = System::Drawing::Size(0, 21);
            this->lblManufactureDate->TabIndex = 4;
            //
            // lblExpiryDate
            //
            this->lblExpiryDate->AutoSize = true;
            this->lblExpiryDate->Location = System::Drawing::Point(170, 340);
            this->lblExpiryDate->Name = L"lblExpiryDate";
            this->lblExpiryDate->Size = System::Drawing::Size(0, 21);
            this->lblExpiryDate->TabIndex = 5;
            //
            // lblStatus
            //
            this->lblStatus->AutoSize = true;
            this->lblStatus->Location = System::Drawing::Point(170, 370);
            this->lblStatus->Name = L"lblStatus";
            this->lblStatus->Size = System::Drawing::Size(0, 21);
            this->lblStatus->TabIndex = 6;
            //
            // lnkFile
            //
            this->lnkFile->AutoSize = true;
            this->lnkFile->Location = System::Drawing::Point(170, 400);
            this->lnkFile->Name = L"lnkFile";
            this->lnkFile->Size = System::Drawing::Size(0, 21);
            this->lnkFile->TabIndex = 7;
            this->lnkFile->LinkClicked += gcnew System::Windows::Forms::LinkLabelLinkClickedEventHandler(this, &TrackMedicineForm::lnkFile_LinkClicked);
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(12, 12);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(184, 48);
            this->btnBack->TabIndex = 8;
            this->btnBack->Text = L"<-- Повернутися назад";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this, &TrackMedicineForm::btnBack_Click);
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label1->Location = System::Drawing::Point(258, 175);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(326, 21);
            this->label1->TabIndex = 9;
            this->label1->Text = L"Введіть код препарату для відстеження";
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
            this->label2->Font = (gcnew System::Drawing::Font(L"Segoe UI", 24, System::Drawing::FontStyle::Bold));
            this->label2->Location = System::Drawing::Point(317, 101);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(188, 45);
            this->label2->TabIndex = 10;
            this->label2->Text = L"DrugGuard";
            //
            // TrackMedicineForm
            //
```

```
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->lnkFile);
            this->Controls->Add(this->lblStatus);
            this->Controls->Add(this->lblExpiryDate);
            this->Controls->Add(this->lblManufactureDate);
            this->Controls->Add(this->lblCode);
            this->Controls->Add(this->lblName);
            this->Controls->Add(this->btnTrack);
            this->Controls->Add(this->txtCode);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"TrackMedicineForm";
            this->Text = L"DrugGuard";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
    #pragma endregion

    private: System::Void btnTrack_Click(System::Object^ sender, System::EventArgs^ e) {
        String^ code = txtCode->Text;

        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT Name, Code, ManufactureDate, ExpiryDate, Status, FilePath FROM Medicines WHERE Code=@code", conn);

        cmd->Parameters->AddWithValue("@code", code);

        SqlDataReader^ reader;

        try {
            conn->Open();
            reader = cmd->ExecuteReader();

            if (reader->Read()) {
                String^ name = reader["Name"]->ToString();
                String^ status = reader["Status"]->ToString();
                String^ manufactureDate = reader["ManufactureDate"]->ToString();
                String^ expiryDate = reader["ExpiryDate"]->ToString();
                String^ filePath = reader["FilePath"]->ToString();

                lblName->Text = "Назва ліків: " + name;
                lblCode->Text = "Код ліків: " + code;
                lblManufactureDate->Text = "Дата виготовлення: " + manufactureDate;
                lblExpiryDate->Text = "Термін придатності: " + expiryDate;
                lblStatus->Text = "Статус: " + status;
                lnkFile->Text = "Сертифікат: " + Path::GetFileName(filePath);
                lnkFile->Tag = filePath;
            }
            else {
                lblName->Text = "Ліки не знайдено.";
                lblCode->Text = "";
                lblManufactureDate->Text = "";
                lblExpiryDate->Text = "";
                lblStatus->Text = "";
                lnkFile->Text = "";
                lnkFile->Tag = nullptr;
            }
        }
        catch (Exception^ ex) {
            lblName->Text = "Помилка підключення до бази даних: " + ex->Message;
            lblCode->Text = "";
            lblManufactureDate->Text = "";
            lblExpiryDate->Text = "";
            lblStatus->Text = "";
            lnkFile->Text = "";
            lnkFile->Tag = nullptr;
        }
        finally {
            if (reader != nullptr) {
                reader->Close();
            }
            conn->Close();
        }
    }

    private: System::Void lnkFile_LinkClicked(System::Object^ sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e) {
```

```
            String^ filePath = lnkFile->Tag->ToString();
            if (!String::IsNullOrEmpty(filePath) && File::Exists(filePath)) {
                System::Diagnostics::Process::Start(filePath);
            }
            else {
                MessageBox::Show("Файл не знайдено.");
            }
        }

        private: System::Void btnBack_Click(System::Object^ sender, System::EventArgs^ e) {
            this->Close();
            previousForm->Show();
        }

        private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
        }
    };
}
```

ViewMedicinesForSaleForm.h

```
#pragma once

namespace MedicationProtectionApp {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
    using namespace System::Data::SqlClient;
    using namespace System::Collections::Generic;

    public ref class ViewMedicinesForSaleForm : public System::Windows::Forms::Form
    {
    public:
        ViewMedicinesForSaleForm(Form^ previousForm)
        {
            InitializeComponent();
            this->previousForm = previousForm;
            medicineCodes = gcnew Dictionary<String^, String^>();
            LoadMedicines();
        }

    protected:
        ~ViewMedicinesForSaleForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::ListBox^ lstMedicines;
    private: System::Windows::Forms::Label^ lblDetails;
    private: System::Windows::Forms::RichTextBox^ rtbDescription;
    private: System::Windows::Forms::PictureBox^ pictureBox;
    private: System::Windows::Forms::Button^ btnBack;
    private: Form^ previousForm;
    private: Dictionary<String^, String^>^ medicineCodes;

    private:
        System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
            this->lstMedicines = (gcnew System::Windows::Forms::ListBox());
            this->lblDetails = (gcnew System::Windows::Forms::Label());
            this->rtbDescription = (gcnew System::Windows::Forms::RichTextBox());
            this->pictureBox = (gcnew System::Windows::Forms::PictureBox());
            this->btnBack = (gcnew System::Windows::Forms::Button());
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox))->BeginInit();
            this->SuspendLayout();
            //
            // lstMedicines
            //
            this->lstMedicines->FormattingEnabled = true;
            this->lstMedicines->ItemHeight = 21;
            this->lstMedicines->Location = System::Drawing::Point(12, 89);
            this->lstMedicines->Name = L"lstMedicines";
            this->lstMedicines->ScrollAlwaysVisible = true;
            this->lstMedicines->Size = System::Drawing::Size(200, 361);
```

```
            this->lstMedicines->TabIndex = 0;
            this->lstMedicines->SelectedIndexChanged += gcnew System::EventHandler(this,
&ViewMedicinesForSaleForm::lstMedicines_SelectedIndexChanged);
            //
            // lblDetails
            //
            this->lblDetails->AutoSize = true;
            this->lblDetails->Location = System::Drawing::Point(230, 12);
            this->lblDetails->Name = L"lblDetails";
            this->lblDetails->Size = System::Drawing::Size(0, 21);
            this->lblDetails->TabIndex = 1;
            //
            // rtbDescription
            //
            this->rtbDescription->Location = System::Drawing::Point(230, 190);
            this->rtbDescription->Name = L"rtbDescription";
            this->rtbDescription->ReadOnly = true;
            this->rtbDescription->ScrollBars = System::Windows::Forms::RichTextBoxScrollBars::Vertical;
            this->rtbDescription->Size = System::Drawing::Size(528, 260);
            this->rtbDescription->TabIndex = 2;
            this->rtbDescription->Text = L"";
            //
            // pictureBox
            //
            this->pictureBox->Location = System::Drawing::Point(558, 25);
            this->pictureBox->Name = L"pictureBox";
            this->pictureBox->Size = System::Drawing::Size(200, 150);
            this->pictureBox->SizeMode = System::Windows::Forms::PictureBoxSizeMode::Zoom;
            this->pictureBox->TabIndex = 3;
            this->pictureBox->TabStop = false;
            //
            // btnBack
            //
            this->btnBack->BackColor = System::Drawing::SystemColors::ActiveCaptionText;
            this->btnBack->FlatStyle = System::Windows::Forms::FlatStyle::Popup;
            this->btnBack->ForeColor = System::Drawing::SystemColors::Control;
            this->btnBack->Location = System::Drawing::Point(12, 12);
            this->btnBack->Name = L"btnBack";
            this->btnBack->Size = System::Drawing::Size(200, 32);
            this->btnBack->TabIndex = 4;
            this->btnBack->Text = L"<--Повернутися назад";
            this->btnBack->UseVisualStyleBackColor = false;
            this->btnBack->Click += gcnew System::EventHandler(this,
&ViewMedicinesForSaleForm::btnBack_Click);
            //
            // ViewMedicinesForSaleForm
            //
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::None;
            this->ClientSize = System::Drawing::Size(796, 521);
            this->Controls->Add(this->btnBack);
            this->Controls->Add(this->pictureBox);
            this->Controls->Add(this->rtbDescription);
            this->Controls->Add(this->lblDetails);
            this->Controls->Add(this->lstMedicines);
            this->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12));
            this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedToolWindow;
            this->Name = L"ViewMedicinesForSaleForm";
            this->Text = L"DrugGuard";
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox))->EndInit();
            this->ResumeLayout(false);
            this->PerformLayout();
```

```
        }
    #pragma endregion

        private: System::Void lstMedicines_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {
            if (lstMedicines->SelectedItem == nullptr) {
                return;
            }

            String^ selectedName = lstMedicines->SelectedItem->ToString();
            String^ code = medicineCodes[selectedName];

            SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-ACME1K0;Database=MedicationSystem;Integrated Security=True;");
            SqlCommand^ cmd = gcnew SqlCommand("SELECT * FROM MedicinesForSale WHERE Code=@code", conn);

            cmd->Parameters->AddWithValue("@code", code);

            SqlDataReader^ reader;

            try {
                conn->Open();
                reader = cmd->ExecuteReader();

                if (reader->Read()) {
                    String^ name = reader["Name"]->ToString();
                    String^ manufacturer = reader["Manufacturer"]->ToString();
                    String^ manufactureDate = reader["ManufactureDate"]->ToString();
                    String^ expiryDate = reader["ExpiryDate"]->ToString();
                    String^ description = reader["Description"]->ToString();
                    String^ imageUrl = reader["FilePath"]->ToString();

                    lblDetails->Text = "Назва: " + name + "\nКод: " + code + "\nДата виробництва: " + manufactureDate + "\nТермін придатності: " + expiryDate;
                    rtbDescription->Text = description;

                    if (!String::IsNullOrEmpty(imageUrl)) {
                        if (Uri::IsWellFormedUriString(imageUrl, UriKind::Absolute)) {
                            System::Net::WebRequest^ request = System::Net::WebRequest::Create(imageUrl);
                            System::Net::WebResponse^ response = request->GetResponse();
                            System::IO::Stream^ stream = response->GetResponseStream();
                            pictureBox->Image = Image::FromStream(stream);
                        }
                        else {
                            pictureBox->Image = Image::FromFile(imageUrl);
                        }
                    }
                    else {
                        pictureBox->Image = nullptr;
                    }
                }
                else {
                    lblDetails->Text = "Препарат не знайдено.";
                    rtbDescription->Text = "";
                    pictureBox->Image = nullptr;
                }
            }
            catch (Exception^ ex) {
                lblDetails->Text = "Помилка підключення до бази даних: " + ex->Message;
                rtbDescription->Text = "";
```

```
            pictureBox->Image = nullptr;
        }
        finally {
            if (reader != nullptr) {
                reader->Close();
            }
            conn->Close();
        }
    }

    private: System::Void btnBack_Click(System::Object^ sender, System::EventArgs^ e) {
        this->Close();
        previousForm->Show();
    }

    private: System::Void LoadMedicines()
    {
        SqlConnection^ conn = gcnew SqlConnection("Server=DESKTOP-
ACME1K0;Database=MedicationSystem;Integrated Security=True;");
        SqlCommand^ cmd = gcnew SqlCommand("SELECT Name, Code FROM MedicinesForSale", conn);

        SqlDataReader^ reader;
        medicineCodes = gcnew Dictionary<String^, String^>();

        try {
            conn->Open();
            reader = cmd->ExecuteReader();

            while (reader->Read()) {
                String^ name = reader["Name"]->ToString();
                String^ code = reader["Code"]->ToString();
                lstMedicines->Items->Add(name);
                medicineCodes[name] = code;
            }
        }
        catch (Exception^ ex) {
            MessageBox::Show("Помилка підключення до бази даних: " + ex->Message);
        }
        finally {
            if (reader != nullptr) {
                reader->Close();
            }
            conn->Close();
        }
    }
    };
}
```