НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

До захисту допущено:

Завідувач кафедри

_____ Надія БУРАУ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та системи навігації і керування»

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: «Автоматизована система зчитування інформації»

Виконав:
студент IV курсу, групи ПГ-91
Обруснік Андрій Віталійович _____

Керівник:

Доцент кафедри КІОНС, к.т.н., доцент Півторак Д.О.
_____

Рецензент:
Доцент, к.т.н., доцент Шевченко В.В.
_____ _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих оптичних та навігаційних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології

Освітня програма - Комп'ютерно - інтегровані технології та системи навігації і керування

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Надія БУРАУ

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Обрусніку Андрію Віталійовичу

1. Тема роботи «Автоматизована система зчитування інформації », керівник роботи Півторак Діана Олександрівна, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи 5 червня 2023 р.

3. Вихідні дані до роботи_ *Автоматизована система повинна давати можливість зчитувати інформацію зі штрих-коду*

4. Зміст роботи *1. Огляд стану проблеми. 2. Огляд сучасних засобів створення автоматизованих систем. 3. Реалізація та тестування готового продукту.*

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презентація _____

6. Консультанти розділів роботи*

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

7. Дата видачі завдання 3 травня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд стану проблеми	22.05.2023 р.	
2.	Огляд сучасних засобів створення автоматизованих систем.	29.05.2023 р.	
3.	Реалізація та тестування готового продукту	03.06.2023 р.	
10.	Оформлення пояснювальної записки. Підготовка до захисту	05.06.2023 р.	

Студент Андрій ОБРУСНІК

Керівник Діана ПІВТОРАК

Анотація

У даному дипломній роботі було розроблено автоматизовану систему зчитування інформації а саме мобільний додаток, який забезпечує сканування штрих-кодів та автоматичний пошук відповідних записів у Google Sheets. Додаток надає зручний спосіб швидкого доступу до інформації, пов'язаної з кожним штрих-кодом. Основна функціональність додатку полягає у скануванні штрих-кодів з використанням камери мобільного пристрою та пошуку відповідних записів у базі даних, яка зберігається у Google Sheets. Знайдені записи безпосередньо відображаються в додатку, де користувач може редагувати, видаляти або додавати нові дані за допомогою штрих-коду. Цей мобільний додаток надає зручний та ефективний спосіб керування інформацією, пов'язаною з продуктами або об'єктами, які мають штрих-коди. Він забезпечує швидкий доступ до даних, спрощує їх редагування та дозволяє зручно додавати нові записи за допомогою штрих-коду.

Ключові слова: Штрих-код, мобільний додаток, Xamarin, C#, Google Sheets

Abstract

This diploma thesis presents the development of an automated information retrieval system, specifically a mobile application that enables barcode scanning and automatic search of corresponding records in Google Sheets. The application provides a convenient way to quickly access information associated with each barcode. The main functionality of the application involves scanning barcodes using the mobile device's camera and searching for corresponding records in a database stored in Google Sheets. The found records are displayed directly in the application, where users can edit, delete, or add new data using the barcode. This mobile application offers a convenient and efficient means of managing information related to products or objects with barcodes. It facilitates fast data access, simplifies editing, and allows convenient addition of new records using barcodes.

Keywords: Barcode, mobile application, Xamarin, C#, Google Sheets

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

XML — eXtensible Markup Language	Розширювана мова розмітки
SOAP — Simple Object Access Protocol	Простий протокол доступу до об’єктів
REST — Representational State Transfer	Передача стану представлення
XAML — eXtensible Application Markup Language	Розширювана мова розмітки додатків
IDE — Integrated Development Environment	Інтегроване середовище розроблення
API — Application programming interface	Прикладний програмний інтерфейс
JSON — JavaScript Object Notation	Позначення об’єктів JavaScript
HTTP — HyperText Transfer Protocol	Протокол передачі гіпертексту
HTTPI — HyperText Transfer Protocol Secure	Захищений протокол передачі гіпертексту
LINQ — Language Integrated Query	Мовний інтегрований запит

ВСТУП

У даній роботі розглядається тема «Автоматизована система зчитування інформації». В сучасному світі інформаційні технології відіграють важливу роль у бізнес-процесах, тому автоматизація збору та обробки інформації є ключовою задачею для багатьох організацій. Дана робота має на меті створення автоматизованої системи зчитування інформації, яка дозволить швидко та ефективно збирати та обробляти дані.

У роботі розкривається актуальність теми та обґрунтовування її значення для бізнесу та інших сфер діяльності. Описуються проблеми, які виникають при ручному зборі та обробці інформації, а також недоліки існуючих систем автоматизації.

Актуальність теми «Автоматизована система зчитування інформації» полягає в тому, що в сучасному світі інформаційні потоки стали дуже великими та їх важко контролювати, тому автоматизовані системи зчитування інформації є необхідними для ефективної роботи в різних галузях діяльності. Такі системи дозволяють отримувати потрібну інформацію швидко та точно, що дозволяє економити час і зусилля, знижувати витрати та підвищувати якість роботи.

Ручний збір інформації має свої проблеми, які стають ще більш помітними при великій кількості даних. Один з основних недоліків полягає в тому, що процес збору інформації може бути дуже часомістким та неефективним. До того ж, людина може допустити помилки під час збору даних, що може призвести до неточностей та недостовірності відповідної інформації.

Крім того, ручний збір інформації може бути дуже часо- та ресурсозатратним процесом, особливо якщо потрібно збирати великі обсяги даних. В результаті цього можуть виникати проблеми з часом виконання та точністю зібраної інформації.

Отже, створення автоматизованої системи зчитування інформації є актуальним завданням, яке може значно полегшити процес збору та аналізу

даних. Використання такої системи може знизити ризики помилок та забезпечити більш точні та надійні результати.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ЛІТЕРАТУРИ ЗА ТЕМОЮ “АВТОМАТИЗОВАНА СИСТЕМА ЗЧИТУВАННЯ ІНФОРМАЦІЇ”

1.1 Поняття інформації

Походження терміна «інформація» пов'язане з латинським словом «informatio», що означає роз'яснення, виклад, відомості. Незважаючи на широке використання цього слова, поняття інформації є предметом активних дискусій у наукових колах. В даний час наука намагається знайти загальні властивості та закономірності, що описують багатогранне поняття інформації. Проте, це поняття залишається частково інтуїтивним і має різні відтінки значень в різних сферах людської діяльності [1].

Науково точну кількість інформації, що міститься у конкретних текстах, фресках або генетичному коді людини, складно визначити. Наукова спільнота не надає чіткої відповіді на це питання, і дослідження в цій області продовжуються. Однак, у теорії інформації були розроблені підходи до вимірювання кількості інформації. Згідно з теорією інформації Клода Шеннона, одиницею інформації є біт, який представляє себе як кількість інформації, необхідну для розрізнення двох рівноймовірних повідомлень. Таким чином, біт використовується для вимірювання кількості інформації, яка передається в повідомленні. Існують підходи до визначення кількості інформації, що базуються на математичних поняттях ймовірності та логарифма. Ці підходи дозволяють оцінити кількість інформації, яку містить повідомлення, з точки зору новизни або зменшення невизначеності наших знань про об'єкт. Важливо зазначити, що кількість інформації може бути виміряна лише в рамках певних умов і контексту. Різні групи даних можуть мати різну кількість інформації, залежно від їх характеристик і властивостей. Отже, хоча точне вимірювання кількості інформації є складною задачею, теорія інформації надає підходи та

одиниці вимірювання, які дозволяють оцінювати та порівнювати кількість інформації в різних контекстах [1].

В обчислювальній техніці біт використовується як найменша одиниця пам'яті, яка представляє знак «0» або «1». Біт є досить малим для вимірювання інформації, тому на практиці використовується більша одиниця виміру - байт. Байт складається з восьми бітів і може закодувати один з 256 символів алфавіту комп'ютерної клавіатури. Крім байта, широко використовуються ще більші похідні одиниці інформації, такі як кілобайт, мегабайт, гігабайт, терабайт і петабайт. Вони використовуються для виміру великих обсягів оброблюваної інформації. Але на практиці найбільш поширеними є двійкові одиниці, такі як біт і байт, для вимірювання та обробки інформації в комп'ютерах [1].

Інформаційні ресурси представляють собою накопичені ідеї та вказівки, які можуть бути відтворені і реалізовані. Ці ресурси включають книги, статті, патенти, дисертації, науково-дослідну документацію, дослідно-конструкторські матеріали, технічні переклади, передовий виробничий досвід та багато іншого. Відмінність інформаційних ресурсів від інших видів ресурсів, таких як праця, енергія, мінерали і т.д., полягає в тому, що інформаційні ресурси збільшуються швидше, коли їх використовують більше. Інформаційна технологія охоплює методи і пристрої, які люди використовують для обробки інформації. Це включає різноманітні інструменти, програми, системи, алгоритми та інші засоби, які допомагають збирати, зберігати, обробляти, передавати та аналізувати інформацію. Інформаційна технологія є важливим інструментом для розвитку суспільства, науки, бізнесу та багатьох інших галузей [1].

Сотні тисяч років людство займалося обробкою інформації. Спочатку це відбувалося за допомогою рахунків і писемності. Однак, близько п'ятдесяти років тому, з'явилися комп'ютери, що спричинило винятково швидкий розвиток інформаційних технологій. У сучасному розумінні, термін «інформаційна технологія» використовується в контексті використання комп'ютерів для обробки інформації. Інформаційні технології охоплюють широкий спектр обчислювальної техніки, побутової електроніки, телебачення та радіомовлення.

Вони знайшли своє застосування в різних галузях, таких як промисловість, торгівля, банківська сфера, освіта, охорона здоров'я, медицина, транспорт, зв'язок, сільське господарство, соціальне забезпечення та багато інших. Розвинуті країни розуміють важливість вдосконалення інформаційних технологій, хоча це може бути складним та витратним завданням. На сьогоднішній день, створення великомасштабних інформаційно-технологічних систем стало економічно досяжним, що призвело до запуску національних дослідницьких та освітніх програм, спрямованих на сприяння їх розвитку [1].

На рис. 1.1 показано перелік основних дій з інформацією.

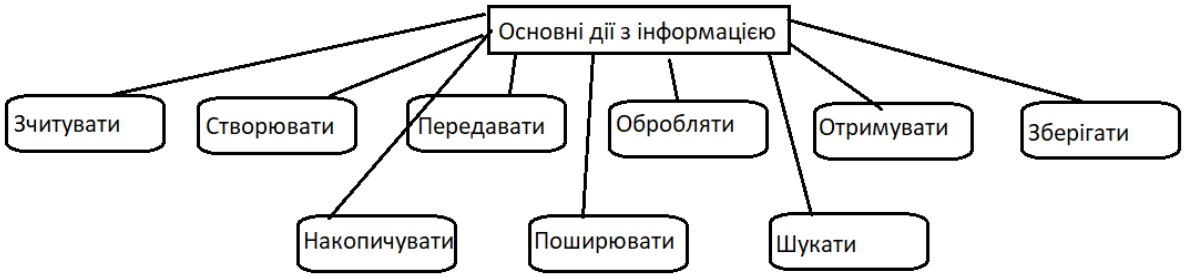

Рис. 1.1 Основні дії з інформацією

Обробка інформації включає в себе виконання алгоритмів для отримання нової інформації з вихідних даних. Цей процес є ключовою операцією, що дозволяє збільшити обсяг та розмаїття інформації. Засоби обробки інформації, зокрема комп'ютери, використовуються для виконання алгоритмів обробки інформації.

Один з важливих процесів обробки інформації - це стиснення даних, яке полягає у зменшенні обсягу інформації без втрати важливих даних або порушення їх цілісності. Це може бути корисним, наприклад, для збереження даних, передачі через обмежені канали зв'язку або оптимізації роботи з даними [1].

Існує кілька методів стиснення інформації, які використовуються в різних сферах [2]:

1. Безвтратне стиснення (Lossless Compression): Цей метод стиснення дозволяє зменшити обсяг даних без втрати якості. Він застосовується там,

де точність та повнота даних є важливими. Приклади безвтратного стиснення включають алгоритми, такі як ZIP, GZIP та PNG.

2. Втратне стиснення (Lossy Compression): Цей метод стиснення дозволяє досягти великого ступеня стиснення, але за рахунок невеликої втрати якості. Він широко використовується для мультимедійних даних, наприклад, зображень, звуку або відео. Приклади втратного стиснення включають формати JPEG для зображень та MP3 для аудіо.
3. Методи стиснення з втратами з контролем якості: Ці методи стиснення дозволяють регулювати ступінь стиснення та втрати якості. Вони корисні, коли потрібно знайти баланс між обсягом даних і якістю. Наприклад, формат JPEG дозволяє встановити різні рівні якості стиснення.
4. Спеціалізовані методи стиснення: У різних галузях застосовуються спеціалізовані методи стиснення, які оптимізовані для конкретних типів даних. Наприклад, для текстових даних можуть використовуватися методи стиснення, що враховують особливості мови або повторюваність символів.

Зчитування інформації є процесом отримання даних з певного джерела або носія та перетворення їх у зрозумілу форму для подальшого використання. Цей процес може включати такі етапи, як розпізнавання, інтерпретацію та обробку даних, щоб забезпечити їх зрозумілість та використання у відповідних контекстах.

Існує багато способів зчитування інформації, включаючи такі:

1. Очні способи зчитування даних: Це найпоширеніший спосіб отримання інформації, який базується на використанні зору. Він включає читання тексту, перегляд зображень, розпізнавання символів та інші операції, які можуть бути виконані за допомогою зору, наприклад, читання книг, перегляд веб-сторінок або перегляд фотографій.
2. Автоматичне зчитування: Цей метод залежить від використання технологій для автоматичного отримання інформації з джерел без безпосередньої участі людини. Наприклад, використання сканерів штрих-

кодів або QR-кодів, оптичного розпізнавання символів (OCR), систем розпізнавання голосу, сенсорів розпізнавання відбитків пальців та інших технологій.

3. Електронне зчитування: Цей спосіб передбачає використання електронних пристроїв, таких як комп'ютери, смартфони або планшети, для зчитування та обробки інформації. Це може бути здійснене через різноманітні програми, веб-сайти, електронні книги, відео- або аудіоматеріали та інші електронні ресурси.

1.2 Штрих-код. Типи. Застосування

Штрих-код — це спосіб кодування інформації, який можна зчитувати різноманітними пристроями. Він складається з чорно-білих смуг та інших геометричних елементів, розташованих у певному порядку за певним алгоритмом. Штрих-коди можна використовувати для кодування будь-якої інформації. Сканери штрих-кодів зчитують зображення та перетворюють їх на читабельну інформацію. Залежно від типу штрих-коду, 1D штрих-коди можуть містити 20-25 символів, тоді як 2D-коди можуть містити до 2000 символів.

Існує кілька поширених типів штрих-кодів

1. Штрих-коди типу EAN/UPC (рис. 1.2): Ці штрих-коди використовуються для ідентифікації товарів у роздрібній торгівлі. EAN (European Article Number) використовується в Європі, а UPC (Universal Product Code) - в США та Канаді. Вони складаються з 12 або 13 цифр [4].

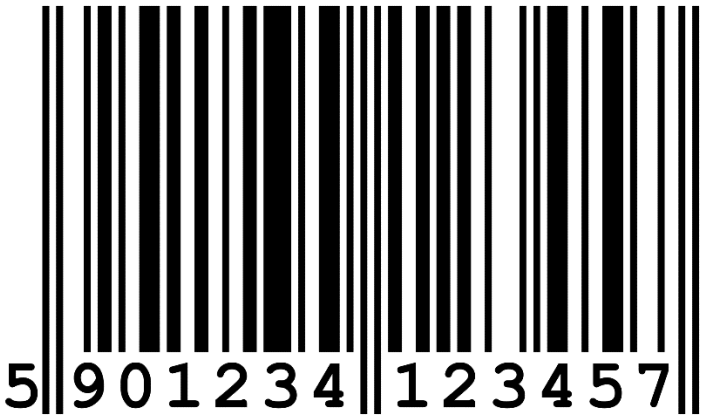

Рис. 1.2 Приклад штрих-коду типу EAN/UPC [4]

2. Штрих-коди типу QR-код (рис. 1.3): QR-коди (Quick Response) - це квадратні штрих-коди, які можуть зберігати більше інформації, включаючи текст, URL-адреси, контактну інформацію та інше. Вони здатні бути зчитані за допомогою смартфонів та інших пристроїв з камерами [5].

Рис. 1.3 Приклад штрих-коду типу QR-код

3. Штрих-коди типу Code 39 (рис.1.4): Цей тип штрих-кодів використовується для різноманітних застосувань, включаючи логістику та ідентифікацію товарів. Вони можуть містити цифри, великі літери та спеціальні символи [7].

Рис. 1.4 Приклад штрих-коду типу Code 39

Штрих-коди мають широке застосування в різних галузях і принципово спрощують ідентифікацію та відстеження товарів і інформації. Ось деякі основні області, де використовуються штрих-коди [6]:

1. Роздрібна торгівля: У сфері роздрібної торгівлі штрих-коди використовуються для швидкого та точного сканування товарів при їх продажу. Вони дозволяють ефективно відстежувати запаси, спрощують процес касового обслуговування і допомагають керувати інформацією про продукти.
2. Логістика та керування ланцюгом постачань: Штрих-коди використовуються для ідентифікації та відстеження товарів протягом всього ланцюга постачання - від виробника до кінцевого споживача. Вони дозволяють точно визначити місцезнаходження товарів, контролювати запаси, відправляти та отримувати товари швидко та ефективно.
3. Керування активами: Штрих-коди використовуються для ідентифікації та відстеження активів, таких як обладнання, інструменти, транспортні засоби тощо. Це допомагає відстежувати рух активів, зберігати інформацію про обслуговування та ремонт, а також полегшує інвентаризацію та керування активами.
4. Медична галузь: У медицині штрих-коди використовуються для ідентифікації пацієнтів, медичного обладнання, медикаментів та інших матеріалів. Вони допомагають уникнути помилок в дозуванні ліків, відстежувати застосування медичних процедур та забезпечувати точність та безпеку у медичних процесах.
5. Керування документами: Штрих-коди використовуються для ідентифікації та відстежування документів, що допомагає впорядковувати та керувати великим обсягом інформації. Вони можуть бути використані для швидкого пошуку та категоризації документів, контролю доступу та збереження історії документів.

Штрих-коди є ефективними інструментами, що сприяють автоматизації та спрощенню процесів в різних галузях. Вони дозволяють швидко та точно ідентифікувати товари та інформацію, спрощують ведення обліку та керування даними, покращують продуктивність та зменшують ризик виникнення помилок.

Штрих-коди зазвичай використовують чорно-білу кольорову схему з кількох причин. Перш за все, таке кольорове оформлення забезпечує максимальний контраст, що дозволяє легко та точно зчитувати штрих-коди. Використання чорного кольору для смуг та білого фону сприяє створенню чіткості та відмінності штрих-коду, що є важливим для його успішного розпізнавання пристроями зчитування.

Друга причина використання чорно-білої схеми полягає в її універсальності. Ця кольорова схема підходить для різних типів сканерів та зчитувачів, які працюють на принципі розрізнення світла та темряви. Чорно-білі штрих-коди ефективно зчитуються на різних пристроях і спрощують процес ідентифікації товарів та збирання даних. Варто зазначити, що, хоча чорно-біла схема є найпоширенішою, також існують кольорові штрих-коди, які використовуються для спеціальних застосувань або кодування додаткової інформації. Комбінації кольорів в кольорових штрих-кодах можуть включати білий, помаранчевий, жовтий, бежевий, червоний колір для підкладки (матеріалу, на який наноситься штрих-код) і чорний, синій, зелений, темно-коричневий колір для самого штрих-коду [10].

На рис.1.5 зображення допустимих комбінацій кольорів для штрих-кода.

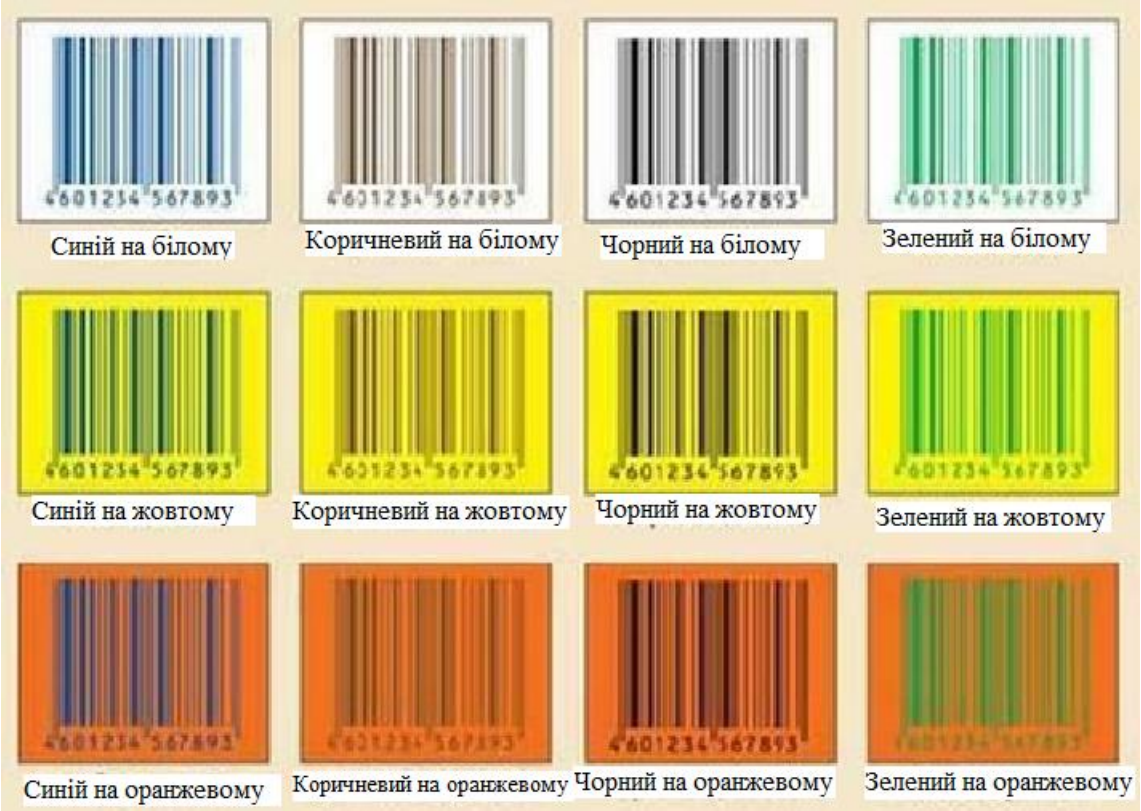

Рис. 1.5 Допустимі комбінації кольорів для штрих-кода[10]

1.3 Кодування штрих-кода

Штрихове кодування є формою ідентифікаційного маркування, яке містить службову інформацію, не маючи безпосередньої користі для споживача. Щодо штрих-кодів, серед покупців існує кілька поширених помилкових тлумачень. Одне з найпоширеніших уявлень полягає в тому, що за першими двома-трьома цифрами штрих-коду можна визначити країну походження товару.

На практиці, перші дві цифри в штрих-коді вказують на національну організацію, в якій зареєстровано підприємство-виробника. Це не обов'язково відображає країну походження товару, оскільки підприємство може бути зареєстровано в одній країні, а його товар може бути вироблений або постачений з іншої країни. Також варто зазначити, що власник товарного знаку (бренду) має переважне право на нанесення штрих-коду, а необхідність його використання може виникнути відповідно до угоди між різними сторонами, які відповідають за виробника, постачальника та інші аспекти товару.

Що стосується інформації, яку можна отримати з штрих-коду, то він не містить конкретних даних про фасон, колір, розмір, строк придатності та інші споживчі властивості товару. Штрих-код може бути використаний для визначення країни реєстрації та перевірки автентичності товару. Вся додаткова інформація про товар зберігається в електронному каталозі виробника, до якого зазвичай доступу немає для покупця. Стандартний штрих-код, що використовується міжнародною організацією EAN International, складається з 13 цифр, з яких перші дві вказують на код країни, наступні п'ять - на код підприємства-виробника або продавця, далі йдуть п'ять цифр для ідентифікації товару, а остання цифра використовується для контролю правильності сканування. На рис.1.6 наведено роз’яснення стандарту EAN International [8].

Рис. 1.6 Роз’яснення стандарту EAN International

Для перевірки автентичності товару виконайте такі кроки: спочатку візьміть всі цифри, які знаходяться на парних позиціях у штрих-коді, потім перемножте отриману суму на три. Далі, візьміть цифри з непарних позицій (крім контрольної цифри) і додайте їх до потроєної суми, що була отримана раніше. Від результату відкиньте першу цифру і відніміть отриману решту від десяти. Остання цифра, яка залишиться, є контрольною цифрою. Якщо ця контрольна цифра не співпадає з останньою цифрою у штрих-коді, то це може свідчити про підробку товару [8].

Зазвичай код країни, вказаний в штрих-коді, призначається Міжнародною асоціацією EAN. Важливо відзначити, що споживачам варто звернути увагу на

те, що код країни в штрих-коді ніколи не буде складатися з однієї цифри. Іноді може статися, що код, нанесений на етикетку, не відповідає зазначеній на упаковці країні виробника. Це може мати кілька причин, і необхідно бути уважним при перевірці штрих-коду та інформації про походження товару (рис.1.7) [8].

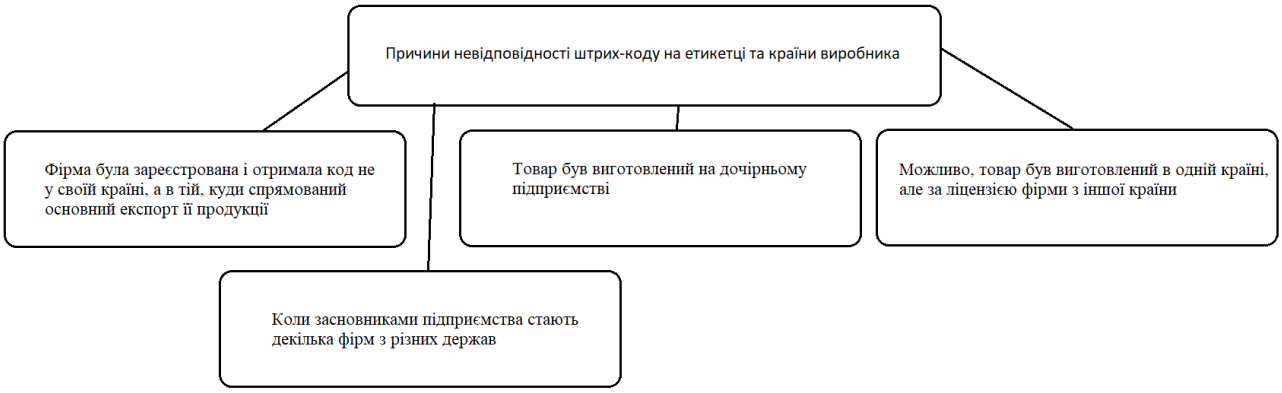

Рис. 1.7 Причини невідповідності штрих-коду на етикетці та країни виробника

1.4 Задача розроблення мобільних додатків, які зчитують штрих-код

Задача розроблення мобільних додатків, які зчитують штрих-коди, є важною і цікавою для багатьох сфер діяльності. Ці додатки використовують можливості камери смартфона або планшета для сканування штрих-кодів, що відкриває широкі можливості для автоматизації та оптимізації процесів.

Однією з основних причин розроблення мобільних додатків для зчитування штрих-кодів є спрощення процесу ідентифікації товарів та збір даних. Завдяки швидкому та точному скануванню штрих-кодів користувачі можуть швидко отримати інформацію про товар, включаючи назву, ціну, опис, сертифікати, дату виробництва та багато іншого. Це спрощує процес купівлі, складання замовлень, контроль за запасами та інші операції [9].

Мобільні додатки, що зчитують штрих-коди, також мають широке застосування в логістиці та керуванні ланцюгом постачань. Вони допомагають відстежувати рух товарів, контролювати запаси, підтримувати актуальну інформацію про поставки та забезпечувати точність та ефективність керування.

Це особливо важливо в галузях, де велика кількість товарів має бути відстежена та оброблена в реальному часі.

Зчитування штрих-кодів через мобільні додатки також застосовується в керуванні активами, документами та в інших сферах. Відстеження активів, ідентифікація документів та збереження інформації стає більш ефективним завдяки можливості швидкого та точного сканування штрих-кодів.

Розроблення мобільних додатків для зчитування штрих-кодів відкриває безліч можливостей для автоматизації, оптимізації та покращення різних процесів. Вона дозволяє користувачам швидко отримувати інформацію про товари та об'єкти, спрощує керування запасами, підвищує ефективність логістичних операцій і покращує загальну продуктивність в багатьох галузях [9].

1.5 Додатки які використовують технологію штрих-коду

В сучасному цифровому світі штрих-коди стали невід'ємною частиною нашого повсякденного життя. Технологія штрих-коду дозволяє швидко та точно ідентифікувати товари та об'єкти, а також збирати та обробляти зв'язану з ними інформацію. В результаті, багато мобільних додатків стали використовувати цю технологію з метою поліпшення функціональності та зручності для користувачів.

Один з найбільш поширених типів додатків, які використовують технологію штрих-коду, це мобільні додатки електронної комерції. Завдяки вбудованій функціональності сканування штрих-кодів, користувачі можуть зручно та швидко здійснювати покупки безпосередньо зі своїх мобільних пристроїв. Вони можуть просто сканувати штрих-код товару, який бачать у магазинах або навіть у своєму оточенні, і здійснювати покупку без необхідності вводити довгий перелік товарів. Це робить процес покупок більш зручним та ефективним для користувачів.

Крім того, існують інші додатки, які використовують технологію штрих-коду з різних сфер діяльності. Наприклад, додатки керування домашньою

бібліотекою, які дозволяють користувачам зберігати каталог своїх книг за допомогою сканування штрих-кодів ISBN. Це дозволяє автоматично отримувати деталі про книги та зручно впорядковувати свою бібліотеку. Додатки супермаркетів також використовують технологію штрих-коду, дозволяючи клієнтам сканувати штрих-коди продуктів для отримання детальної інформації про них та складання списків покупок. Навіть у медичній сфері, штрих-коди використовуються для ідентифікації медичних препаратів, лікарських засобів та пацієнтів, забезпечуючи точність та безпеку в медичному обслуговуванні.

Використання технології штрих-коду в мобільних додатках відкриває безліч можливостей для автоматизації, оптимізації та покращення різних процесів. Вона спрощує покупки, поліпшує керування запасами, дозволяє швидко отримувати інформацію про товари та підвищує продуктивність в різних сферах. З ростом популярності смартфонів і мобільних пристроїв, застосування штрих-кодів у додатках стає все більш широким і різним [11]. Список додатків які вже використовують функцію сканування штрих-кода:

1. «Barcode Scanner» - Цей додаток дозволяє користувачам миттєво сканувати штрих-коди з використанням камери свого смартфона. Він простий у використанні і надає інформацію про товар, включаючи ціну, опис, рейтинги тощо. Це особливо корисно для покупців, які хочуть порівняти ціни або отримати додаткові відомості перед покупкою.
2. «Inventory Tracker» - Цей додаток дозволяє бізнесам вести облік запасів, використовуючи штрих-коди. Він дозволяє сканувати штрих-коди товарів для швидкого і точного запису вхідної та вихідної інформації, відстежувати кількість товарів на складі та автоматично оновлювати запаси.
3. «Price Comparison» - Цей додаток допомагає споживачам порівнювати ціни на товари, використовуючи штрих-коди. Він дозволяє сканувати штрих-коди товарів і порівнювати їхні ціни в різних магазинах або

онлайн-платформах. Користувачі можуть знайти найкращу пропозицію та заощадити гроші при покупці.

4. «Food Tracker» - Цей додаток дозволяє користувачам відстежувати інформацію про продукти харчування за допомогою штрих-кодів. Він надає детальну інформацію про склад продукту, калорійність, алергени, термін придатності та інші характеристики. Це особливо корисно для людей з дієтологічними обмеженнями або хто хоче здійснювати більш обізнаний вибір щодо їжі.
5. «Attendance Tracker» - Цей додаток використовує штрих-коди для ведення обліку присутності на робочому місці або в навчальному закладі. Він дозволяє сканувати штрих-коди на ідентифікаційних картках або квитках, що дозволяє точно відстежувати, коли людина прибуває та відходить. Це зручний і ефективний спосіб автоматизувати процес обліку присутності.

Пристрої для зчитування та обробки штрих-кодів поділяються на:

1. Дротові сканери штрих-кодів (рис.1.8).

Рис. 1.8 Приклад дротових сканерів штрих-кодів

2. Бездротові сканери штрих-кодів (рис. 1.9).

Рис. 1.9 Приклад бездротових сканерів штрих-коду

3. Термінали збору даних та накопичувачі штрих-кодів (рис.1.10).

Рис. 1.10. Приклад терміналів збору та накопичувачів штрих-кодів

РОЗДІЛ 2
РОЗРОБЛЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЗЧИТУВАННЯ ІНФОРМАЦІЇ

2.1 Огляд сучасний засобів створення автоматизованих систем

Автоматизовану систему можна створити різними способами. Одним з них є розроблення додатку до мобільного телефону.

Розроблення мобільних додатків - це широка галузь, що містить велику кількість інструментів, бібліотек і фреймворків. Вибір підходящого фреймворку може бути вирішальним для успіху проекту, оскільки він визначає правила створення програмного забезпечення. Фреймворк забезпечує початкову архітектуру, яку необхідно розширювати та змінювати згідно з потребами проекту. Використання певного програмного інтерфейсу чи компоненту може спрощувати супроводження, модернізацію та масштабування проекту, а також дозволяти реалізовувати бізнес-процеси. Рішення, побудовані на фреймворках, зазвичай працюють швидше, безпечніше та ефективніше, ніж самописні системи.

Отже, важливо вибрати правильні інструменти розроблення мобільного програмного забезпечення з підтримкою автономної синхронізації даних у мобільних додатках.

2.1.1 Середовище розроблення

Середовище розроблення є ключовим інструментом для розроблення програмного забезпечення, і вибір правильного середовища може суттєво вплинути на продуктивність та ефективність проекту. Серед найпоширеніших середовищ виділяю такі як: Visual Studio: Visual Studio та JetBrains Rider: Rider - ці середовища розроблення надають розробникам зручні інструменти для написання, тестування, налагодження та впровадження програм на мові. Але

найбільш використовуване середовище розроблення для мови C# та платформи Xamarin є Visual Studio.

Інтегроване середовище розроблення Visual Studio є творчим стартовим майданчиком, який можна використовувати для редагування, налагодження та складання коду, а також для публікації програми. В додаток до стандартного редактора та відладчика, що надаються більшістю інтегрованих середовищ розроблення, Visual Studio включає компілятори, засоби завершення коду, графічні конструктори та багато інших функцій для покращення процесу розроблення програмного забезпечення, і тому основним середовищем розроблення було обрано сучасне середовище розроблення Microsoft Visual Studio [12].

Ця лінійка продуктів включає інтегроване середовище розроблення програмного забезпечення, а також інші інструментальні засоби. Вона дозволяє розробляти різні типи додатків, включаючи консольні додатки, ігри та програми з графічним інтерфейсом, веб-сайти та веб-додатки, а також веб-служби, що працюють на різних платформах. Microsoft Visual Studio підтримує Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight, а також Unix-подібні операційні системи, такі як MacOS і Linux [11]. На рис.2.1 показано інтерфейс середовища розроблення Visual Studio.

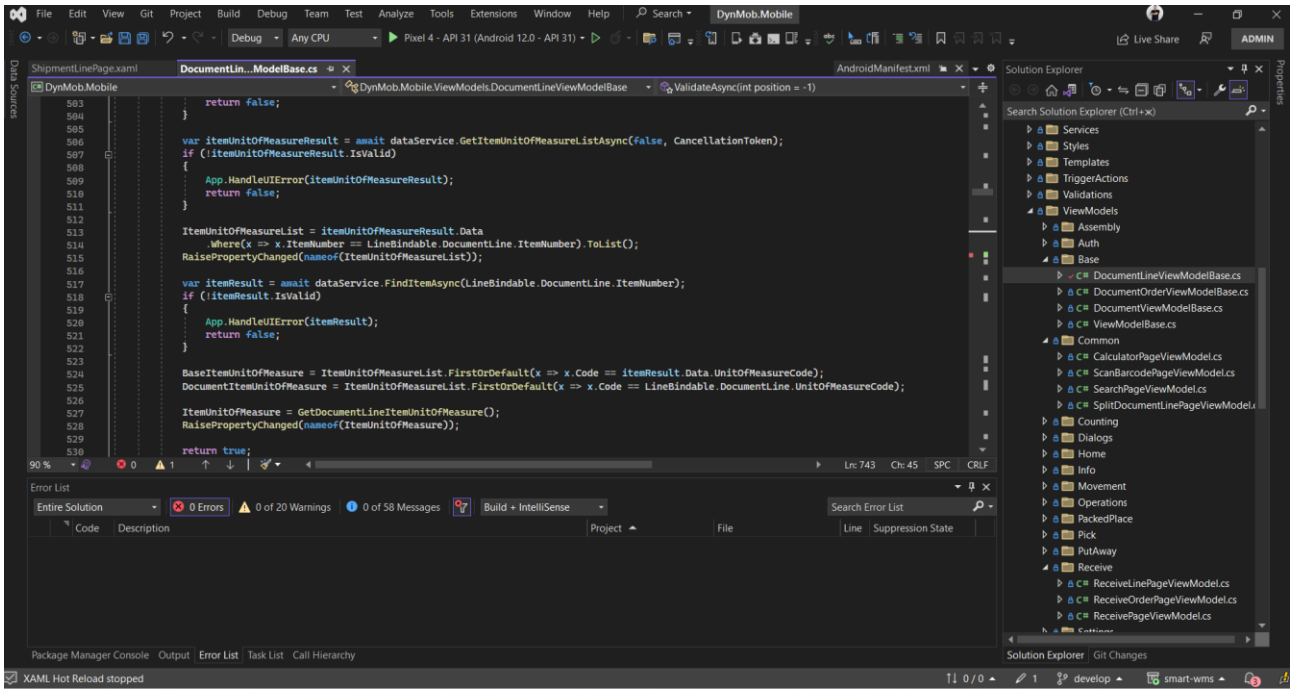

Рис. 2.1 Інтерфейс середовища розроблення Visual Studio

У Microsoft Visual Studio, є редактор вихідного коду з підтримкою технології IntelliSense та можливістю простого рефакторингу коду, вбудований наладчик, який може працювати як на рівні вихідного коду, так і на рівні машинного коду. Вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми баз даних. Visual Studio дозволяє створювати та підключати сторонні додатки (плагіни), щоб розширити функціональність, такі як підтримка систем контролю версій вихідного коду та інструменти для інших аспектів розроблення програмного забезпечення.

Перевагами Visual Studio є підтримка мов програмування C / C ++, C# і Python, зручний редактор коду, інструменти для швидкого редагування макета, вбудований емулятор ОС Android AVD, наявність бібліотеки готових шаблонів та компонентів Android, інтеграція з хмарним сервісом Firebase Service, інтеграція з GitHub і Subversion (SVN), вбудована програма для аналізу вмісту APK та інструменти для тестування додатків. Недоліком є потреба в потужному комп'ютері для швидкої роботи в програмі, але з початку версії Visual Studio 2017, програмний продукт є модульним, що дозволяє встановлювати тільки необхідні модулі для розроблення мобільних додатків, що прискорює роботу

програми. Visual Studio 2019 є лідером серед подібних програмних продуктів у світі розроблення програмного забезпечення [12].

2.1.2 Мова програмування

Серед широкого спектра мов програмування, які використовуються сьогодні, декілька мають особливу популярність і широке застосування. Деякі з найпопулярніших мов програмування включають:

1. Java є однією з найпоширеніших мов програмування, особливо в контексті розроблення програмного забезпечення для мобільних пристроїв та веб-додатків. Вона володіє потужними функціональними можливостями та великою спільнотою розробників.

2. Python здобуває все більшу популярність завдяки своїй простоті, лаконічності та розширюваності. Вона широко використовується у сферах науки про дані, штучного інтелекту, веб-розроблення та автоматизації.

3. JavaScript є мовою програмування, що застосовується в основному для розроблення веб-додатків. Вона дозволяє створювати інтерактивні елементи на веб-сторінках та забезпечує великий вибір фреймворків та бібліотек для розроблення веб-додатків.

4. C# є мовою яка використовується переважно для розроблення програмного забезпечення для платформи .NET.

Однак, C# також є найбільш підходящою мовою програмування для платформи Xamarin. Xamarin є фреймворком для розроблення мобільних додатків, який дозволяє розробникам використовувати спільний код на C# для створення додатків для Android та iOS. Це означає, що розробники, володіючи навичками програмування на C#, можуть використовувати їх для створення мобільних додатків для обох платформ, що значно спрощує розробку та підтримку додатків для різних пристроїв. Крім того, C# пропонує багато інструментів, бібліотек та фреймворків, які полегшують розроблення мобільних додатків на платформі Xamarin і забезпечують високу продуктивність та якість розробленого програмного забезпечення.

C# (C Sharp) - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона була представлена у 2000 році разом із платформою .NET Framework і з того часу стала дуже популярною серед програмістів. C# є однією з основних мов програмування для розроблення програмного забезпечення для платформи Microsoft .NET [14].

C# має схожий синтаксис з C++, але має значно більшу підтримку об'єктно-орієнтованого програмування. Вона дозволяє розробникам створювати різні типи програм, від малих консольних інструментів до великих веб-додатків і гри. C# також має широкі можливості для взаємодії з базами даних, створенням графічних інтерфейсів користувача і багато іншого.

C# є мовою з високою продуктивністю і дуже надійною. Вона підтримує розподілену розробку, що дозволяє розробникам працювати в команді над великими проектами. Крім того, C# має багато вбудованих засобів для тестування коду, що дозволяє програмістам швидко та ефективно відлагоджувати свої програми.

Однією з головних переваг C# є те, що вона поєднує в собі елементи різних мов програмування, таких як C++, Java, Visual Basic і Delphi. Це робить її зручною мовою для тих, хто вже володіє досвідом програмування в інших мовах.

Узагальнюючи, мова програмування C# є потужним інструментом для створення різноманітних програмних продуктів. Вона володіє чіткою і структурованою синтаксичною конструкцією, що сприяє легкому зрозумінню коду для програмістів різного рівня кваліфікації. Ця мова має широкі можливості в області об'єктно-орієнтованого програмування, дозволяє ефективно взаємодіяти з базами даних, мережевими протоколами і веб-сервісами, підтримує розроблення мобільних додатків на платформі Android. Крім того, мова C# є основою для створення програмних продуктів на платформі .NET, що дозволяє програмістам використовувати існуючі бібліотеки, щоб значно скоротити час розроблення. Завдяки відкритому коду та активній спільноті розробників, мова C# є постійно розвиваючимся

інструментом, який пропонує нові можливості для розроблення програмного забезпечення в різних галузях.

2.1.3 Платформа Xamarin

Xamarin - це платформа з відкритим вихідним кодом, яка дозволяє створювати продуктивні додатки для iOS, Android та Windows, використовуючи платформу .NET. Вона пропонує абстракційний рівень, який забезпечує зв'язок між спільним кодом та базовим кодом платформи. Платформа працює в керованому середовищі та забезпечує зручності, такі як виділення пам'яті та збір сміття.

Засіб Xamarin дозволяє розробникам переносити в середньому 90% своїх програм між платформами, використовуючи одну мову програмування або повторне використання наявного коду програми, щоб досягти природніх характеристик, вигляду та відчуття на кожній платформі [13].

Програми Xamarin можна розробляти на персональних комп'ютерах або Mac, після чого вони компілюються в рідні пакети програм, такі як .apk-файли на Android або .ipa-файли на iOS.

Інструмент Xamarin.Forms призначений для створення кроссплатформенних додатків для Android, iOS та Windows 10. Це дає можливість розробникам писати програми більш ніж для однієї платформи, зменшуючи труднощі, пов'язані з відмінністю підходів до побудови графічного інтерфейсу та різних API-розбіжностей. Засоби розроблення для кожної платформи вимагають використання різних мов програмування та середовищ, що може впливати на терміни розроблення та витрати. Xamarin дозволяє уникнути цих проблем, забезпечуючи зручність та ефективність розроблення.
Переваги використання Xamarin.Forms:

— у процесі розроблення створюється єдиний код для всіх платформ;

— інструмент Xamarin надає прямий доступ до Native API кожної платформи;

— при створенні додатків можна використовувати платформу .NET і мову програмування C # (а також F #), яка є досить продуктивною, і в той самий час ясною і простою для освоєння і застосування;

— інструмент Xamarin Forms підтримує кілька платформ. Основні платформи: Android, iOS, UWP, Tizen. Додаткові платформи в стані бета-тестування: MacOS, WPF, GTK# [13].

2.1.4 Google Sheets

Найбільш зручний спосіб представлення великого масиву даних - це використання таблиць. Таблиці є потужним інструментом для організації, структурування та візуалізації великої кількості інформації. Вони дозволяють розміщувати дані у вигляді рядків і стовпців, що спрощує пошук, фільтрацію та сортування даних. Таблиці також підтримують різноманітні форматування, що дозволяє виділяти важливі дані, додавати формули для автоматичного розрахунку значень та створювати графіки для ілюстрації трендів і залежностей. Крім того, таблиці можна легко експортувати та імпортувати з інших програм, що сприяє обміну даними і спільній роботі з колегами та командою. Таким чином, таблиці є універсальним інструментом, що дозволяє ефективно представляти, аналізувати та керувати великими масивами даних.

Одним із різновидів таблиць є Google Sheets – це є веб-програма, яка дозволяє користувачам створювати, оновлювати та редагувати електронні таблиці, а також ділитися даними в режимі реального часу в Інтернеті [19].

Продукт Google надає типові функції електронних таблиць, такі як додавання, видалення та сортування рядків і стовпців. Але, на відміну від інших програм електронних таблиць, Google Sheets дозволяє кільком користувачам, які знаходяться в різних місцях, одночасно спільно працювати над таблицею та спілкуватися через вбудовану систему миттєвих повідомлень. Користувачі можуть завантажувати електронні таблиці безпосередньо зі своїх комп'ютерів або мобільних пристроїв. Програма автоматично зберігає кожну

зміну, і користувачі можуть бачити зміни інших користувачів в режимі реального часу.

Google Sheets входить до складу безкоштовного набору веб-програм Google Docs Editors. Цей набір також включає Google Docs, Google Slides, Google Drawings, Google Forms, Google Sites та Google Keep [19].

На рис. 2.2 зображено інтерфейс таблиці в Google Sheets.

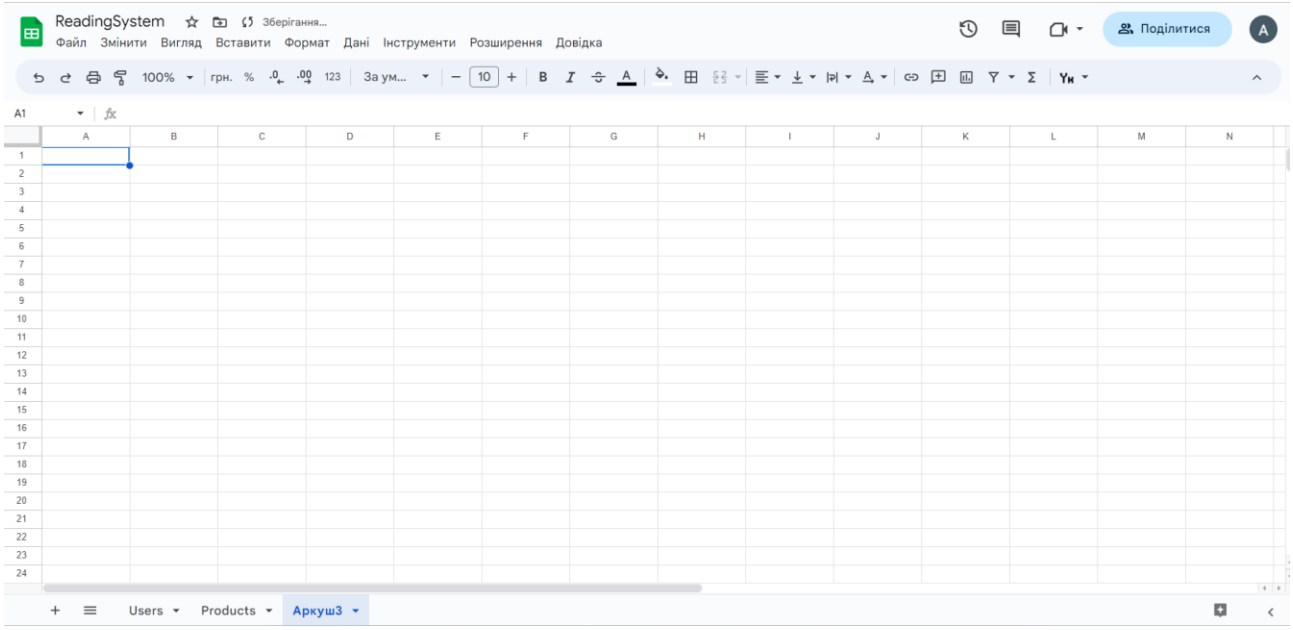

Рис. 2.2 Інтерфейс Google Sheets таблиці

Google Sheets включає такі основні функції [19]:

1. Редагування та форматування електронних таблиць. Це включає операції та функції для введення даних; підсумку даних; перекладу тексту; імпорту даних; перевірки даних; захисту даних; очищення тексту від недрукованих символів; обрізки для видалення пробілів, які можуть бути на початку, в кінці або повторюватися в тексті; фільтрації даних за умовами, такими як дата, алфавітний або числовий порядок; власних умов та теплових карт, які використовують кольори для відображення щільності точок даних у таблиці; а також базових і складних формул.

2. Візуалізація даних. Користувачі можуть створювати графіки, діаграми та інші типи діаграм з даними електронних таблиць та вставляти їх на веб-сайти.

3. Функції на основі машинного навчання. Функція «1» використовує машинне навчання для побудови діаграм, створення зведених таблиць та відповідей на питання про дані. Вона може автоматично оновлюватися на основі вибраних даних.

4. Редагування в автономному режимі. Навіть у випадку відсутності підключення до Інтернету, можна вносити зміни в таблиці Google Sheets в автономному режимі, а зміни оновляться після відновлення з'єднання з Інтернетом.

5. Сумісність. Документи Sheets сумісні з різними форматами, включаючи Excel (XLS), Apache OpenOffice, PDF, текст, HTML та значення, розділені комами (CSV).

6. Інтеграція з продуктами Google. Google Sheets може бути інтегрована з іншими сервісами Google, такими як Drawing, Finance, Form та Translate. Вона також сумісна з файлами Microsoft та має багато спільних клавіатурних скорочень.

7. Функції спільної роботи. Користувачі можуть отримувати повідомлення електронною поштою про коментарі або зміни, зроблені іншими співробітниками у спільній таблиці, і можуть переглядати історію версій.

8. Безпека. Користувачі можуть керувати дозволами на редагування, завантаження, копіювання або друк для конкретних співробітників шляхом надання доступу на рівні окремих осіб, груп або доменів [19].

2.1.5 AppCenter

AppCenter - це платформа для розроблення та керування мобільними додатками, розроблена компанією Microsoft. Вона надає інструменти та сервіси, що допомагають командам розробників створювати, тестувати, розгортати та відстежувати продуктивність своїх мобільних додатків для платформ Android та iOS [20].

Основний функціонал AppCenter включає такі можливості[20]:

1. Кодовий репозиторій: AppCenter забезпечує інтеграцію з репозиторіями Git, такими як GitHub та Bitbucket, для автоматичного виявлення змін в коді та запуску процесу збирання (build) додатку.

2. Збирання та розгортання: AppCenter надає інфраструктуру для збирання (build) додатків, автоматичного створення білдів та розгортання на пристрої або в магазини додатків (наприклад, Google Play Store або App Store). Це дозволяє розробникам швидко та автоматично випускати нові версії своїх додатків.

3. Тестування: AppCenter має вбудовані інструменти для автоматизованого тестування мобільних додатків. Розробники можуть створювати та виконувати тестові сценарії, перевіряти працездатність, виявляти помилки та забезпечувати якість своїх додатків перед релізом.

4. Аналітика: AppCenter надає інструменти для збору та аналізу даних про використання додатків. Розробники можуть отримувати статистику про активність користувачів, краші, виконані дії та інші метрики, що допомагають вдосконалювати функціональність та взаємодію з додатком.

5. Розповсюдження бета-версій: AppCenter дозволяє розробникам розповсюджувати бета-версії своїх додатків серед обмеженого кола користувачів для збору відгуків та виявлення проблем перед публічним випуском.

6. Цикл життя додатку: AppCenter надає засоби для відстеження та керування життєвим циклом додатку, включаючи реєстрацію помилок, пропозиції нових функцій, керування версіями та оновленнями [21].

AppCenter є потужним інструментом для розробників мобільних додатків, який спрощує та автоматизує багато аспектів розроблення та керування додатками, дозволяючи швидше випускати високоякісні продукти на ринок.

На рис. 2.3 показано інтерфейс веб-застосунку платформи AppCenter.

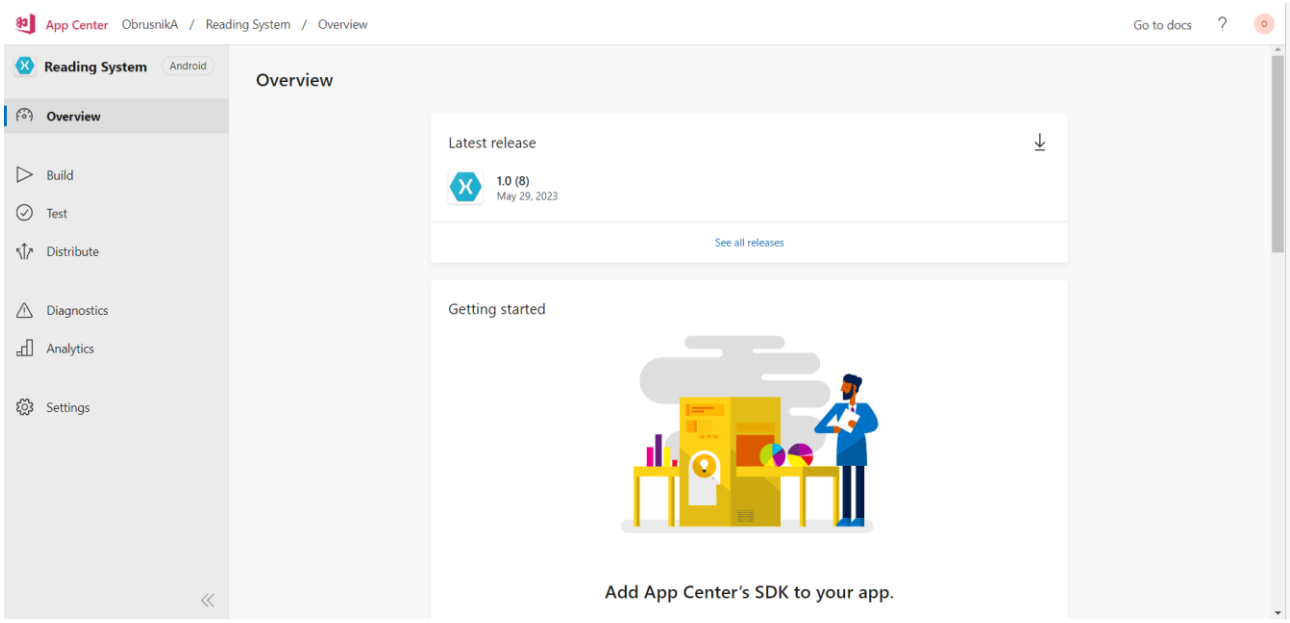

Рис. 2.3 Інтерфейс платформи AppCenter

2.1.6 Система контролю версіями Git

Система контролю версій Git (англ. Git Version Control System) є розподілена система керування версіями файлів і каталогів. Вона розроблена Лінусом Торвальдсом у 2005 році для керування розробкою ядра Linux, а згодом стала популярною серед розробників програмного забезпечення. Git дозволяє відстежувати зміни, які здійснюються у файловій системі, та зберігати їх у вигляді «1» - фіксацій стану файлів у певний момент часу. Він зберігає ці коміти в спеціальній структурі даних, називаній «1». Кожен розробник може мати локальну копію репозиторію, з якої він може взаємодіяти з системою контролю версій [22].

Основні переваги Git [23]:

1. Розподілена система: Кожен розробник має повну копію репозиторію, що дозволяє працювати навіть без з'єднання з мережею. Зміни можуть бути синхронізовані між локальними та віддаленими репозиторіями.

2. Швидкість та продуктивність: Git працює швидко навіть з великими проектами. Він ефективно керує змінами, гілками, злиттями та історією комітів.

3. Гнучкість гілок: Git надає потужні можливості гілкування, що дозволяє розробникам створювати, комбінувати та керувати гілками розроблення незалежно одна від одної.

4. Відновлення та відкат змін: Git зберігає повну історію змін, що дозволяє легко відновити попередні версії файлів або відкотити зміни до попереднього стану.

5. Спільна робота: Git дозволяє ефективно співпрацювати над проектами в команді. Розробники можуть об'єднувати свої зміни, вирішувати конфлікти та коментувати зміни інших учасників [23].

Git є однією з найпопулярніших систем контролю версій у світі розроблення програмного забезпечення та забезпечує потужні інструменти для керування кодом та спільної роботи розробників.

2.2 Розроблення автоматизованої системи зчитування інформації

2.2.1 Підключення до Google Sheets

Для роботи з Google Sheets API в додатках було створено клас GoogleSheetsHelper і він є власним допоміжним класом. Цей клас надає функціональність для отримання автентифікаційних даних, створення об'єкту SheetsService та зручний доступ до сервісу Google Sheets. Основна функціональність класу включає:

Автентифікація: Клас GoogleSheetsHelper забезпечує отримання автентифікаційних даних, необхідних для підключення до Google Sheets API. Ці дані включають тип облікового запису, проект, приватний ключ, електронну адресу клієнта та інші необхідні поля.

Ініціалізація сервісу: З використанням отриманих автентифікаційних даних, клас GoogleSheetsHelper створює об'єкт SheetsService, який представляє з'єднання з Google Sheets API. Цей об'єкт дозволяє взаємодіяти з аркушами, зчитувати та записувати дані та виконувати інші операції, пов'язані з Google Sheets.

Зручний доступ до сервісу: Клас GoogleSheetsHelper надає зручний спосіб отримання доступу до об'єкта SheetsService в додатку. Розробники можуть створити екземпляр класу GoogleSheetsHelper і отримати об'єкт SheetsService, який вже налаштований з відповідними автентифікаційними даними. Завдяки класу GoogleSheetsHelper, розробники можуть спростити роботу з Google Sheets API у своїх додатках. Вони можуть легко отримати доступ до сервісу Google Sheets, виконувати операції з аркушами та обробляти дані, що дозволяє їм ефективно використовувати можливості Google Sheets у своїх програмах.

Основні елементи класу GoogleSheetsHelper включають:

Приватні поля: Scopes і ApplicationName - це масив областей доступу та назва додатку, які використовуються для налаштування з'єднання з Google Sheets API.

Властивість Service: Це властивість, яка повертає об'єкт SheetsService. Цей об'єкт використовується для взаємодії з Google Sheets API, такі як отримання аркушів, зчитування та запис даних тощо.

Конструктор GoogleSheetsHelper(): Цей конструктор ініціалізує об'єкт SheetsService. Він створює об'єкт GoogleCredential з автентифікаційними даними, отриманими з GetCredentialJson(). Потім встановлює створений об'єкт GoogleCredential та назву додатку властивостях HttpClientInitializer і ApplicationName об'єкта BaseClientService.Initializer, який передається при створенні об'єкта SheetsService.

Метод GetCredentialJson(): Цей метод повертає рядок JSON, який містить автентифікаційні дані для підключення до сервісу Google Sheets API. Ці дані включають інформацію про тип облікового запису, проект, приватний ключ, електронну адресу клієнта та інші необхідні поля.

На рис.2.4 показано код класу GoogleSheetsHelper.

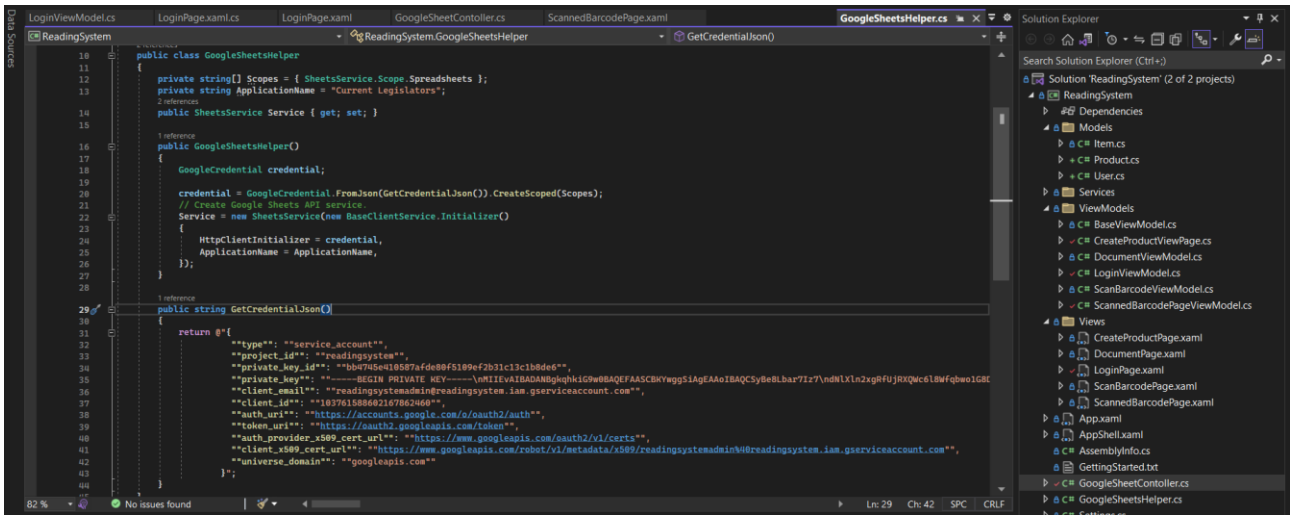

Рис. 2.4 Код класу GoogleSheetsHelper

2.2.2 Програмний інтерфейс для взаємодії з Google Sheets

Для взаємодії з Google Sheets API (Application programming interface) в додатку було створено клас GoogleSheetController. Він містить різноманітні методи, які дозволяють отримувати, оновлювати та видаляти дані з Google Sheets.

GoogleSheetController містить такі методи:

Метод GetLoginData - дозволяє отримати список користувачів з аркуша «1» у Google Sheets. Він використовує об'єкт SpreadsheetsResource.ValuesResource, отриманий з об'єкта GoogleSheetsHelper, для виконання запиту до Google Sheets API і отримання даних. Результатом є список об'єктів типу User, що містять дані про логіни та паролі користувачів.

Метод GetProductsData дозволяє отримати список продуктів з аркуша «1» у Google Sheets. Він працює аналогічно до методу GetLoginData, але отримує дані про продукти. Результатом є список об'єктів типу Product, що містять дані про продукти.

Метод UpdateProduct дозволяє оновити інформацію про продукт в Google Sheets. Він приймає об'єкт типу Product та індекс рядка, який потрібно оновити. За допомогою об'єкта SpreadsheetsResource.ValuesResource та методу Update, дані про продукт оновлюються у відповідному рядку аркуша «1».

Метод DeleteProduct дозволяє видалити інформацію про продукт з Google Sheets. Він приймає індекс рядка, який потрібно видалити. За допомогою об'єкта SpreadsheetsResource.ValuesResource та методу Clear, дані у відповідному рядку аркуша «1» очищаються.

Клас GoogleSheetController дозволяє розробникам легко взаємодіяти з Google Sheets API, отримувати дані користувачів та продуктів, оновлювати ці дані та видаляти їх. Використовуючи цей клас, розробники можуть легко інтегрувати функціональність Google Sheets у свої додатки та зручно працювати з даними збереженими в Google Sheets.

На рис.2.5 – 2.7 показано код класу GoogleSheetController.

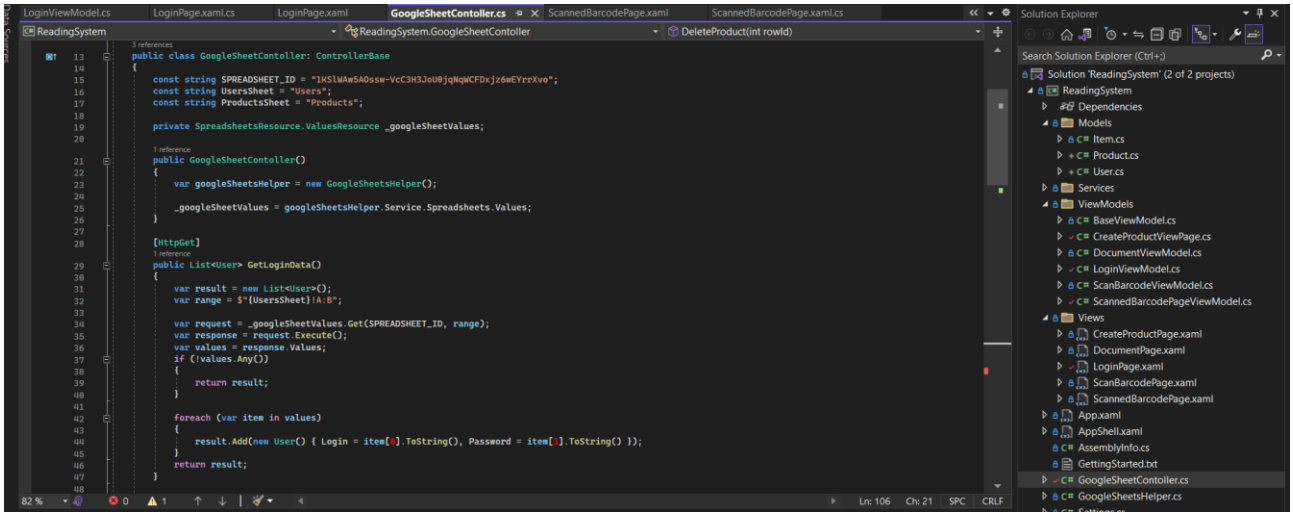

Рис. 2.5 Код конструктора класу GoogleSheetController та код метода GetLoginData

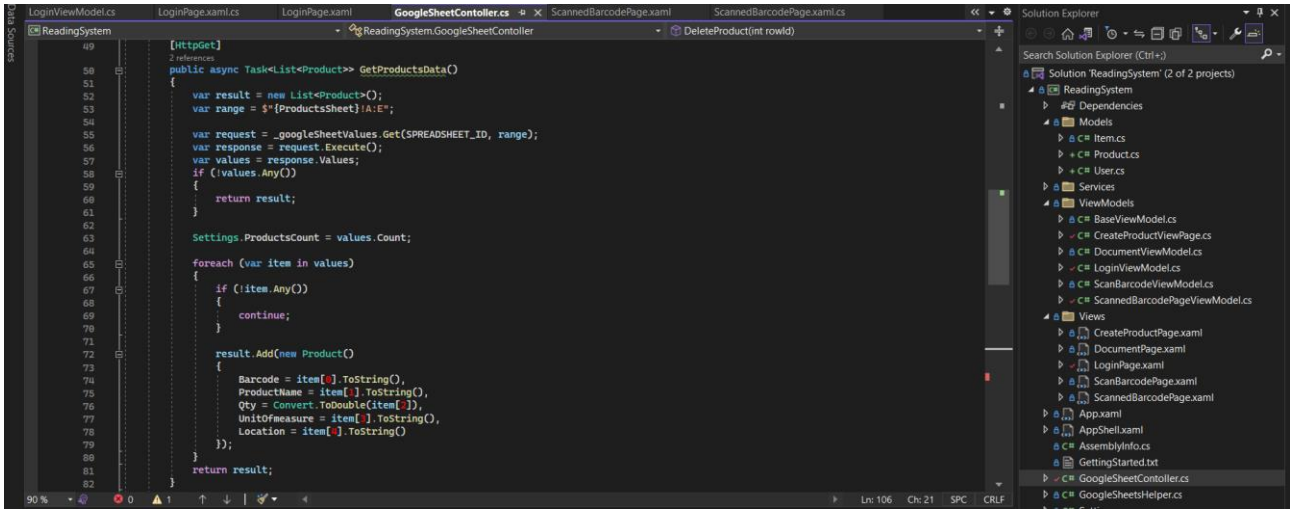

Рис. 2.6 Код метода GetProductsData

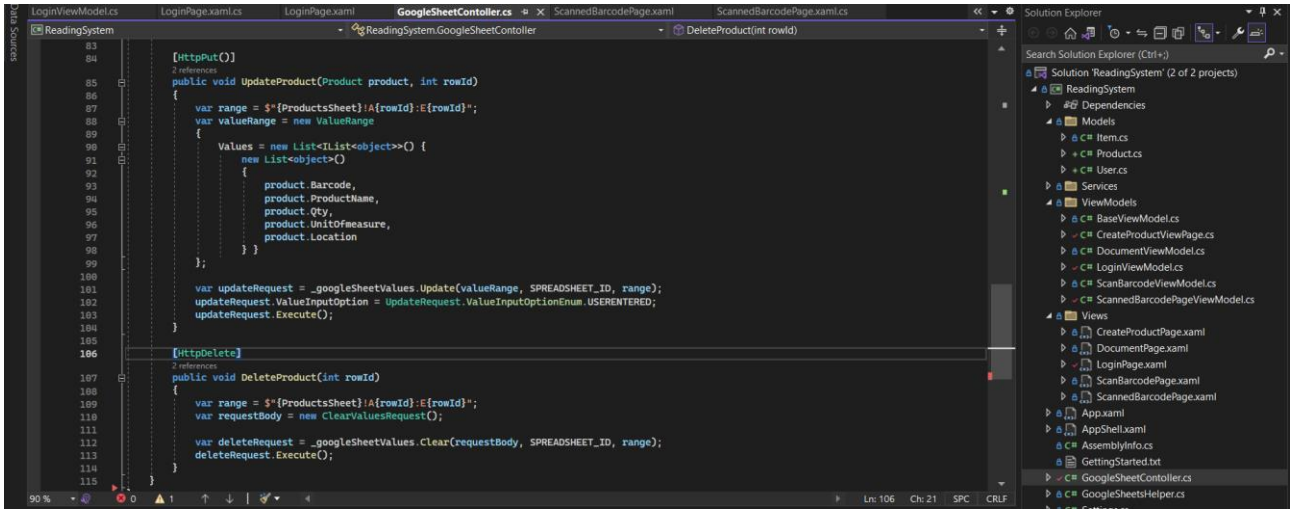

Рис. 2.7 Код метода UpdateProduct та DeleteProduct

2.2.3 Сторінка авторизації

Для входу у додаток було створено сторінку авторизації відповідно до патерну Model-View-ViewModel (MVVM), яка складається з двох компонентів (файлів) ViewModel та View.

ViewModel є окремим класом і має назву LoginViewModel і використовується для зберігання та керування даними, пов'язаними з процесом входу користувача в додаток. LoginViewModel успадковується від BaseViewModel та має 2 властивості User та Password, які і представляють інформацію про користувача та пароль та зберігаються у статичному класі Settings. У конструкторі LoginViewModel значення властивостей User та Password ініціалізуються значеннями з Settings якщо до цього вони вже були заповненні. На рис.2.8 показано код класу LoginViewModel.

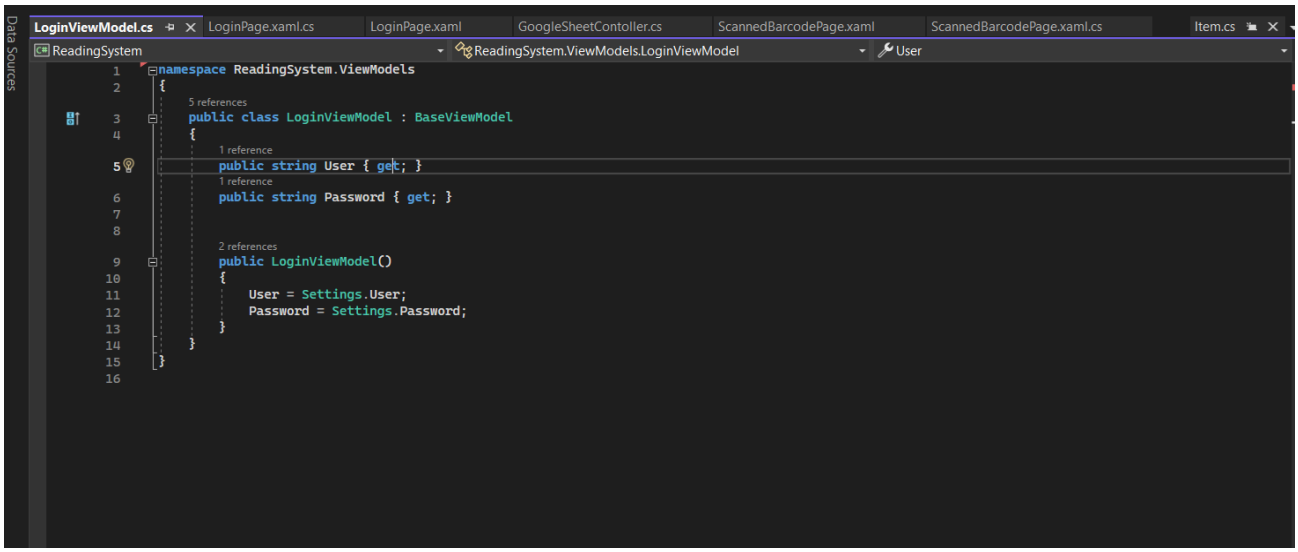

Рис. 2.8 Код класу LoginViewModel

View даної сторінки складається з двох взаємопов’язаних файлів а саме: LoginPage.xaml та LoginPage.xaml.cs де LoginPage.xaml є файлом розмітки (XAML) для сторінки входу (login page) в додатку. Він визначає структуру та вигляд користувацького інтерфейсу сторінки входу, а LoginPage.xaml.cs є кодовим файлом, пов'язаним з розміткою сторінки входу (LoginPage.xaml). Цей файл містить клас LoginPage, який виконує логіку пов'язану зі сторінкою входу.

Логіка, яка присутня в файлі LoginPage.xaml.cs, відповідає за обробку подій та взаємодію з класом LoginViewModel для сторінки входу. Основна логіка включає наступні елементи:

1. Конструктор LoginPage: Цей конструктор викликається при створенні об'єкту сторінки входу. Він ініціалізує екземпляр класу LoginViewModel і встановлює його як контекст прив'язки для розмітки сторінки.

2. Метод LoginClicked: Цей метод викликається при натисканні на кнопку «1». У цьому методі виконується перевірка введеного користувачем логіну і пароля. За допомогою об'єкта LoginViewModel, отримуються дані входу з Google Sheets і порівнюються зі значеннями, введеними користувачем. Якщо дані введені неправильно, відображається повідомлення про помилку. У

разі успішного входу, дані користувача зберігаються в налаштуваннях (Settings), і сторінка входу закривається.

3. Імпорт необхідних просторів імен та успадкування від ContentPage: В коді також імпортуються необхідні простори імен, такі як ReadingSystem.ViewModels, для використання класу LoginViewModel. Клас LoginPage успадковує клас ContentPage з бібліотеки Xamarin.Forms, що дозволяє створити сторінку з вмістом.

На рис.2.9 показано код класу LoginPage.xaml.

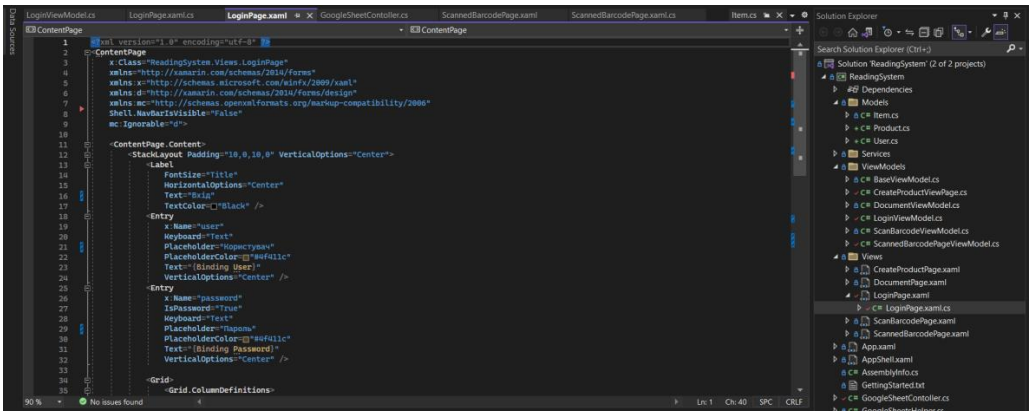

Рис. 2.9 Код файлу LoginPage.xaml

На рис.2.10 показано код класу LoginPage.xaml.cs.

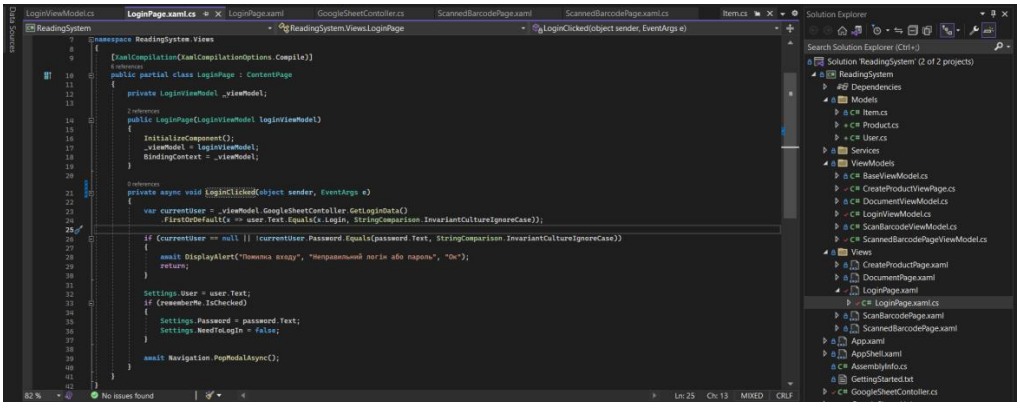

Рис. 2.10 Код класу LoginPage.xaml.cs

На рис.2.11 показано вигляд сторінки авторизації

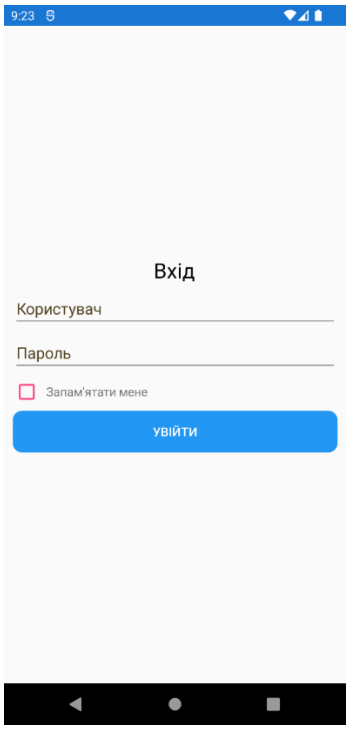

Рисунок 2.11 Екран авторизації в додаток

2.2.4 Сторінка списку існуючих продуктів

Для відображення вже існуючих продуктів, які є в таблиці Google Sheets було створено сторінку відповідно до патерну Model-View-ViewModel (MVVM), яка складається з двох компонентів(файлів) ViewModel та View.

ViewModel виражена класом DocumentViewModel і відповідає за керування логікою та станом користувацького інтерфейсу на сторінці товарів. Даний клас виконує важливу роль у забезпеченні взаємодії між відображенням даних і бізнес-логікою додатку за допомогою своїх властивостей та функціоналу. DocumentViewModel в себе включає:

Властивість логічного типу (bool) IsRefreshing, яка вказує, чи відбувається оновлення даних. Якщо значення IsRefreshing дорівнює true, це означає, що відбувається процес оновлення даних. Властивість реалізована з використанням механізму властивостей та механізму сповіщень INotifyPropertyChanged, щоб повідомляти про зміни значення властивості.

Присутня команда RefreshCommand, яка виконує оновлення даних на сторінці. Вона виконується, коли користувач ініціює процес оновлення,

наприклад, за допомогою жесту «1» або кнопки «1». Ця команда включає асинхронний метод GetProducts, який оновлює дані про продукти.

Метод GetProducts() який є асинхронний і виконується в окремому потоці, завдяки чому не тормозить виконання головної логіки програми. Цей метод виконує запит на отримання даних про продукти з джерела даних (з Google Sheets таблиці). У цьому методі може бути реалізована логіка, пов'язана з отриманням і обробкою даних. Після успішного отримання даних, вони можуть бути збережені у відповідні властивості ViewModel (Products) для відображення на сторінці.

DocumentViewModel має конструктор без параметрів, який викликається при створенні екземпляра ViewModel. У конструкторі можна виконати початкову ініціалізацію, налаштування та підготовку до відображення даних.

DocumentViewModel дозволяє відокремити бізнес-логіку та стан відображення від самого інтерфейсу. Це покращує модульність та розширюваність коду, оскільки логіка додатку може бути протестована окремо від користувальницького інтерфейсу. ViewModel також надає механізм сповіщень, який автоматично оновлює відображення даних при зміні їх стану. На рис.2.12 показано код класу DocumentViewModel.

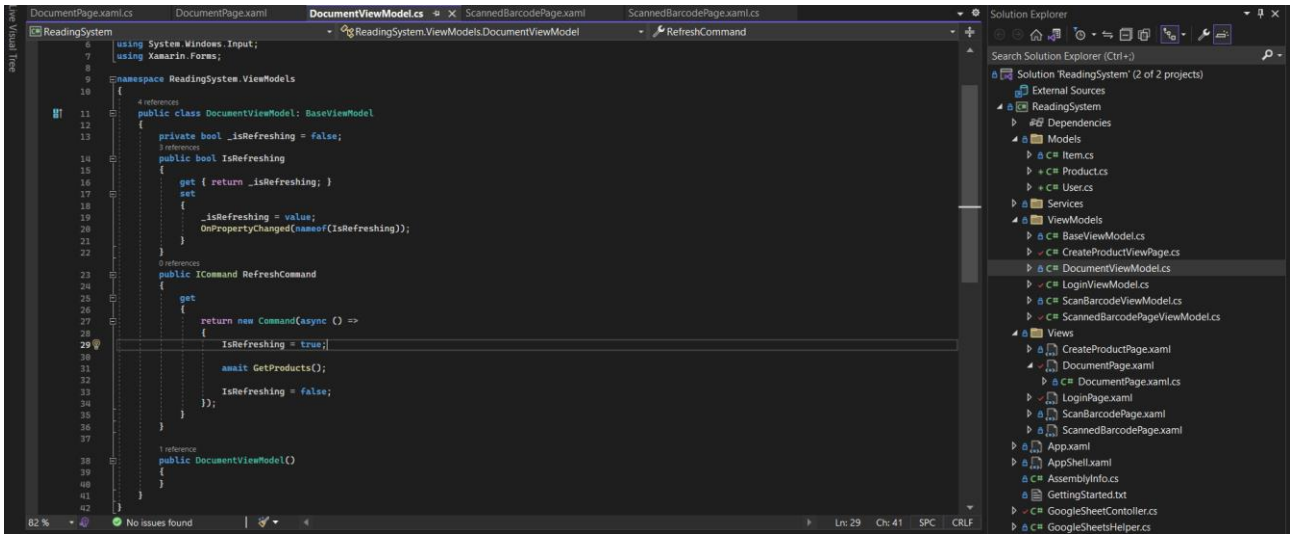

Рис. 2.12 Код класу DocumentViewModel

View даної сторінки складається з двох файлів: файлу DocumentPage.xaml, який є файлом розмітки та DocumentPage.xaml.cs, який є кодовим файлом до файлу розмітки.

Так як DocumentPage.xaml є файлом розмітки він в собі включає декілька основних елементів таких як:

1. ScrollView, щоб забезпечити прокрутку, якщо вміст перевищує розмір екрану.

2. ListView - цей елемент представляє списковий контроль, який відображає колекцію елементів у вигляді списку. Властивість ItemsSource прив'язана до властивості Products у ViewModel, що дозволяє відображати дані про продукти.

3. DataTemplate - цей елемент визначає вигляд одного елемента списку. В даному випадку, він містить <ViewCell>, який представляє один рядок даних в списку. У цьому <ViewCell> є <Grid>, який визначає сітку з колонками та рядками для розміщення елементів даних. Для кожного стовпця в сітці є <Label>, які відображають властивості елемента продукту, такі як назва, розташування, кількість та одиниця виміру.

Файл DocumentPage.xaml визначає розмітку інтерфейсу сторінки документа, що дозволяє відображати список продуктів з відповідними властивостями кожного продукту. Також він містить панель інструментів з кнопкою «1», яка дозволяє вийти з облікового запису.

DocumentPage.xaml.cs є файлом коду C#, який відповідає за логіку та поведінку сторінки продуктів (DocumentPage) у Xamarin.Forms. Основні аспекти та функціонал, реалізований у файлі DocumentPage.xaml.cs, включають:

Клас DocumentPage успадкований від класу ContentPage і представляє код для сторінки документа.

Конструктор DocumentPage(DocumentViewModel documentViewModel): Цей конструктор ініціалізує сторінку документа та приймає об'єкт DocumentViewModel як параметр. Він також встановлює BindingContext

сторінки на цей DocumentViewModel, що дозволяє зв'язувати властивості та команди ViewModel з елементами у розмітці XAML.

LogoutClicked(object sender, EventArgs e): Цей метод викликається при кліку на кнопку «1» у панелі інструментів. Він виконує дії, пов'язані з виходом з облікового запису, наприклад, очищення даних авторизації. Також, він навігує до сторінки входу (LoginPage), щоб користувач міг увійти з новим обліковим записом.

Клас DocumentPage.xaml.cs виконує роль кодової частини, що доповнює розмітку XAML (DocumentPage.xaml), і реалізує взаємодію з елементами сторінки та ViewModel для забезпечення функціональності та коректної поведінки сторінки документа.

На рис.2.13 показано xaml файлу DocumentPage.xaml.

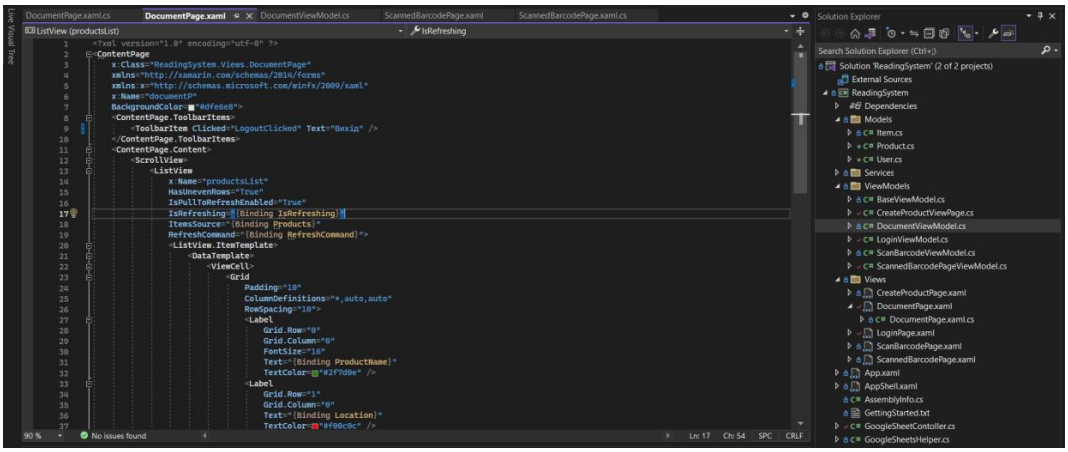

Рис. 2.13 Код файлу DocumentPage.xaml

На рис.2.14 показано код класу DocumentPage.xaml.cs.

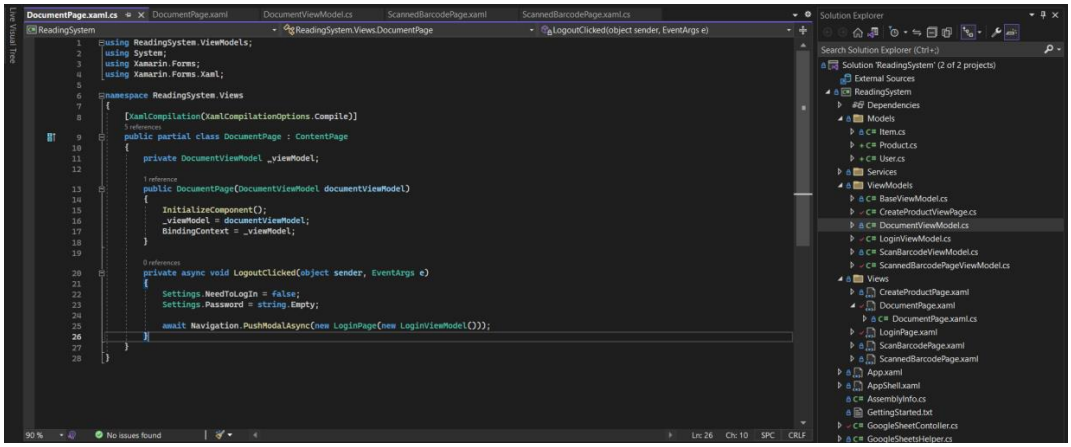

Рис. 2.14 Код файлу DocumentPage.xaml.cs

На рис.2.15 показаний вигляд сторінки списку товарів.

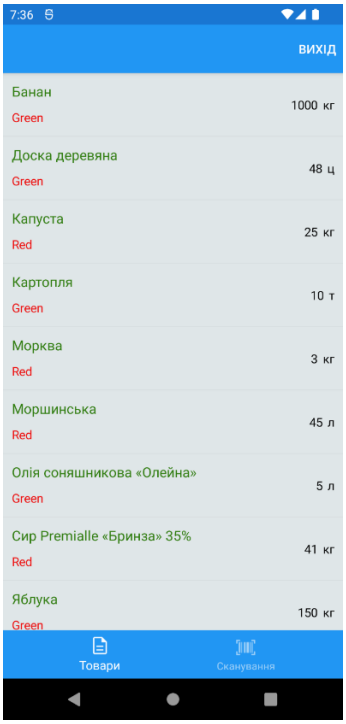

Рис. 2.15 Екран сторінки списку товарів

2.2.5 Сторінка сканування

Основною метою додатку є ефективна обробка інформації, зокрема редагування, додавання та видалення товарів. Для цього була розроблена сторінка сканування, яка відповідає за основну обробку даних. Однак, з метою полегшення процесу додавання товарів, було вирішено виділити його в окрему підсторінку для зручності. Редагування та видалення товарів залишилися на основній сторінці. Відповідно до архітектурного патерну MVVM (Model-View-ViewModel), для сторінки редагування та видалення була створений елемент ViewModel - ScannedBarcodePageViewModel. Цей елемент ViewModel відповідає за керування даними та бізнес-логікою, пов'язаною з редагуванням та видаленням товарів. Також було створено відповідний файл View - ScannedBarcodePage, який відображає інтерфейс користувача для взаємодії зі сторінкою редагування та видалення. Для сторінки додавання товару було створено окрему ViewModel - CreateProductViewPage, яка відповідає за керування процесом додавання нового товару. Відповідно, було розроблено файл View - CreateProductPage, який забезпечує відображення відповідного

інтерфейсу користувача для зручного введення даних та додавання нових товарів.

Це розподілення функціональності на різні сторінки та їх відповідні ViewModel та View дозволяє забезпечити більш зручний та інтуїтивно зрозумілий досвід користування розроблювальним мобільним додатком.

Клас, ScannedBarcodePageViewModel, є ViewModel для сторінки редагування та видалення товару. Він містить декілька властивостей та методів для керування даними та бізнес-логікою пов'язаною зі скануванням штрих-коду, пошуком товару за штрих-кодом та виконанням відповідних дій.

Основні властивості цього класу включають:

1. Product: представляє обраний товар, який буде редагуватись або видалятись.

2. IsVisible: вказує, чи є сторінка видимою. Ця властивість може використовуватись для керування відображенням елементів інтерфейсу користувача на сторінці.

3. CurrentRowId: зберігає індекс поточного товару в списку, що дозволяє визначити його позицію при редагуванні або видаленні.

4. Метод GetProductByBarcode, який приймає штрих-код товару як параметр і повертає відповідний товар зі списку товарів. Для цього він використовує метод GetProductsData з GoogleSheetContoller для отримання списку всіх товарів. Потім за допомогою LINQ-запиту знаходить товар, чий штрих-код збігається з переданим значенням. Після знаходження товару, встановлює значення властивості CurrentRowId, яке вказує на його позицію в списку.

В цілому, цей клас відповідає за керування даними і логікою для сторінки редагування та видалення товару. Він забезпечує зручний доступ до вибраного товару та виконує необхідні дії пов'язані зі скануванням штрих-коду та пошуком відповідного товару.

Файл ScannnedBarcodePage.xaml.cs є кодом кодофайлу (code-behind) для XAML-файлу ScannnedBarcodePage.xaml. Цей кодовий файл відповідає за

поведінку сторінки редагування та видалення товару і взаємодію з відповідною ViewModel (ScannedBarcodePageViewModel).

Основні пункти, які можна виділити з коду:

1. У конструкторі ScannnedBarcodePage ініціалізується ViewModel (_viewModel) та прив'язується до контексту прив'язки даних сторінки (BindingContext).

2. В методі OnAppearing перевіряється, чи є значення штрих-коду в налаштуваннях програми. Якщо таке значення існує, то воно встановлюється у полі вводу штрих-коду (barcodeEntry).

3. У методі GoBarcodePageClicked обробляється подія натискання кнопки сканування штрих-коду. Він відкриває модальну сторінку для сканування штрих-коду.

4. У методі ScanBarcodeClicked обробляється подія натискання кнопки «1». Введене значення штрих-коду зчитується з barcodeEntry. Якщо значення не порожнє, викликається метод GetProductByBarcode з ViewModel для пошуку відповідного товару за штрих-кодом. Якщо товар знайдено, властивість IsVisible встановлюється на true, що виключає відображення решти елементів у StackLayout. Якщо товар не знайдено, користувачу пропонується створити новий товар з введеним штрих-кодом.

5. Методи UpdateSheet, ClearLayout і DeleteProduct відповідають за взаємодію з GoogleSheetContoller і виконують відповідні дії з оновленням або видаленням товару з використанням даних з ViewModel. Також вони викликають метод Clear для очищення полів та скидання видимості елементів.

6. Метод barcodeEntry_PropertyChanged обробляє зміни властивості Text в barcodeEntry. Він перевіряє, чи поле штрих-коду не є порожнім і встановлює відповідну видимість кнопки «1».

У цілому, кодовий файл ScannnedBarcodePage.xaml.cs відповідає за обробку подій та взаємодію з ViewModel для сторінки редагування та видалення товару. Він реалізує функціонал введення штрих-коду, сканування, видалення, оновлення та очищення даних товару.

На рис.2.16 показано код класу ScannedBarcodePageViewModel.

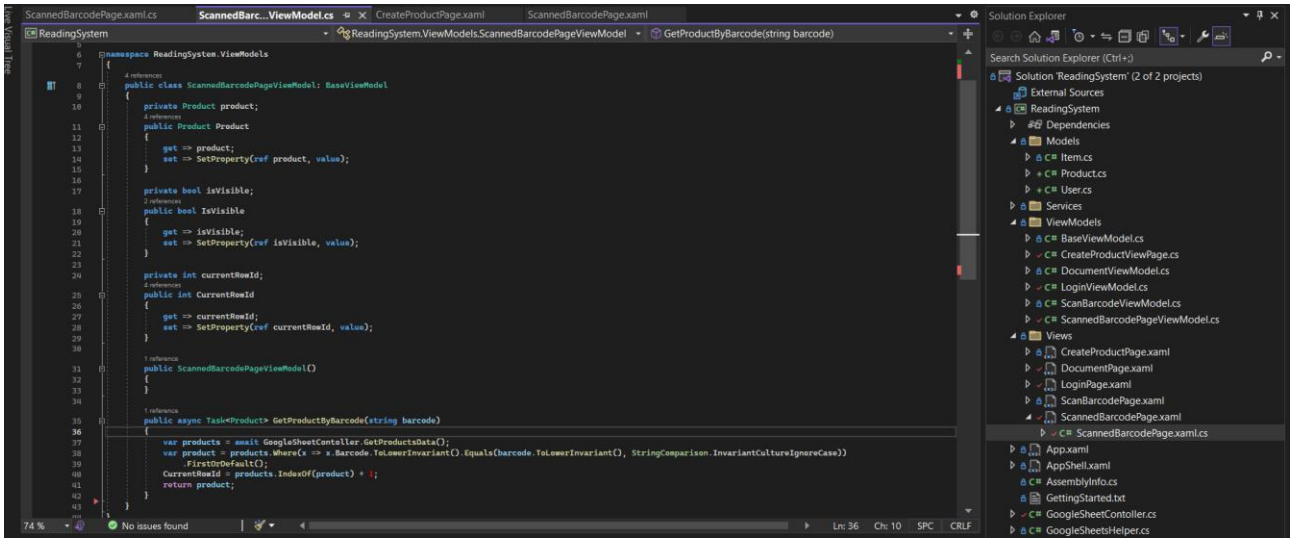

Рис. 2.16 Код класу ScannedBarcodePageViewModel

На рис.2.17 показано код класу ScannnedBarcodePage.xaml.cs.

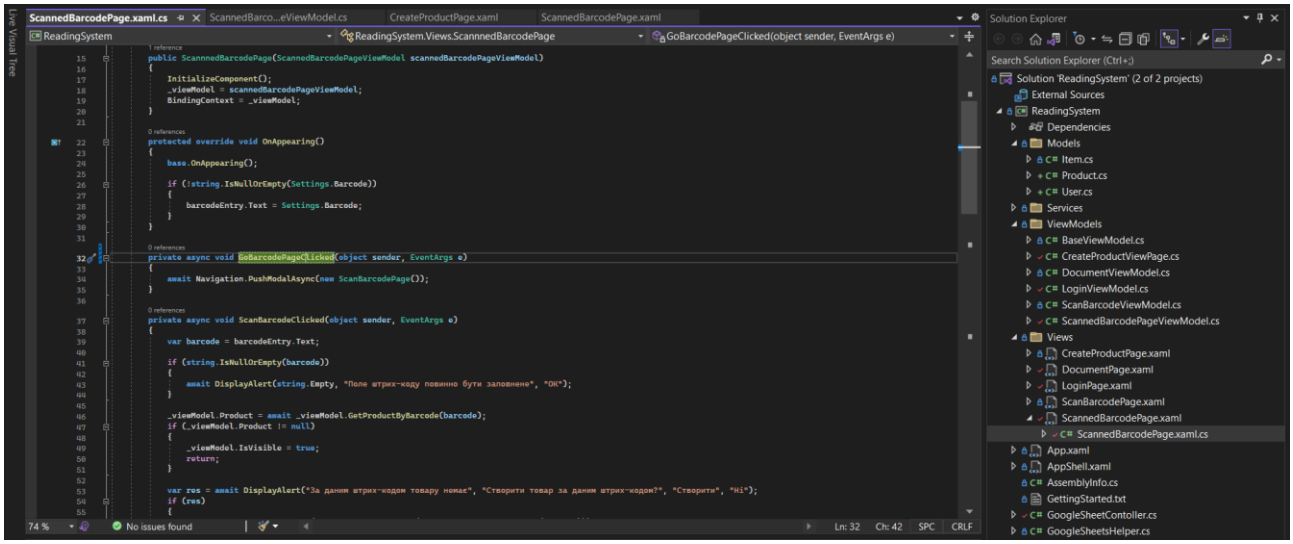

Рис. 2.17 Код файлу ScannnedBarcodePage.xaml.cs

На рис.2.18 показано вигляд сторінки редагування та видалення товару.

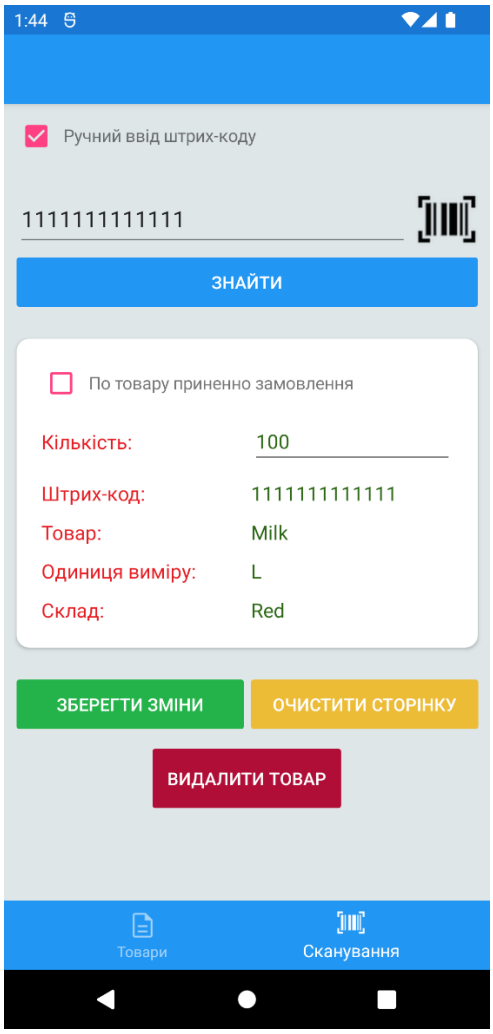

Рисунок 2.18 Екран сторінки редагування та видалення

Клас CreateProductViewPage є ViewModel для створення нового товару. Він містить властивості та методи, необхідні для взаємодії зі сторінкою створення товару.

Основні пункти, які можна виділити з коду:

1. Властивість Product представляє об'єкт типу Product, який містить дані нового товару, які вводяться на сторінці створення.

2. Властивість Barcode відображає штрих-код нового товару. Вона також прив'язана до відповідного поля вводу на сторінці.

3. Конструктор CreateProductViewPage приймає штрих-код як параметр і встановлює відповідне значення властивості Barcode.

4. Метод AddNewProduct викликається при додаванні нового товару. Він оновлює товар у GoogleSheetContoller, передаючи його дані та номер рядка

для оновлення. Також він викликає асинхронний метод GetProducts на головному потоці (Device.InvokeOnMainThreadAsync), який оновлює дані товарів.

У цілому, клас CreateProductViewPage відповідає за збереження нового товару та взаємодію з GoogleSheetContoller. Він надає методи для оновлення та отримання даних товарів, а також зберігає штрих-код нового товару.

Файл CreateProductPage.xaml.cs представляє код XAML-сторінки CreateProductPage разом з відповідним кодом C#.

Основні пункти, які можна виділити з коду:

1. Клас CreateProductPage є кодом відображення сторінки створення нового товару.

2. Конструктор CreateProductPage приймає об'єкт типу CreateProductViewPage в якості параметра. Він ініціалізує внутрішню змінну viewModel і встановлює її як контекст прив'язки.

3. Метод AddProduct викликається при натисканні кнопки «1». Він створює новий об'єкт типу Product, використовуючи дані, введені на сторінці. Далі, викликає асинхронний метод AddNewProduct з екземпляром viewModel для додавання нового товару.

4. Після успішного додавання товару, викликається метод DisplayToastAsync для відображення сповіщення про успішне додавання.

5. Завершується метод AddProduct викликом PopAsync для повернення до попередньої сторінки.

У цілому, цей файл відображає сторінку створення нового товару та взаємодіє з відповідним CreateProductViewPageViewModel, передаючи дані товару для додавання та повідомляючи про результат додавання за допомогою сповіщення.

На рис.2.20 показано код класу CreateProductViewPage.

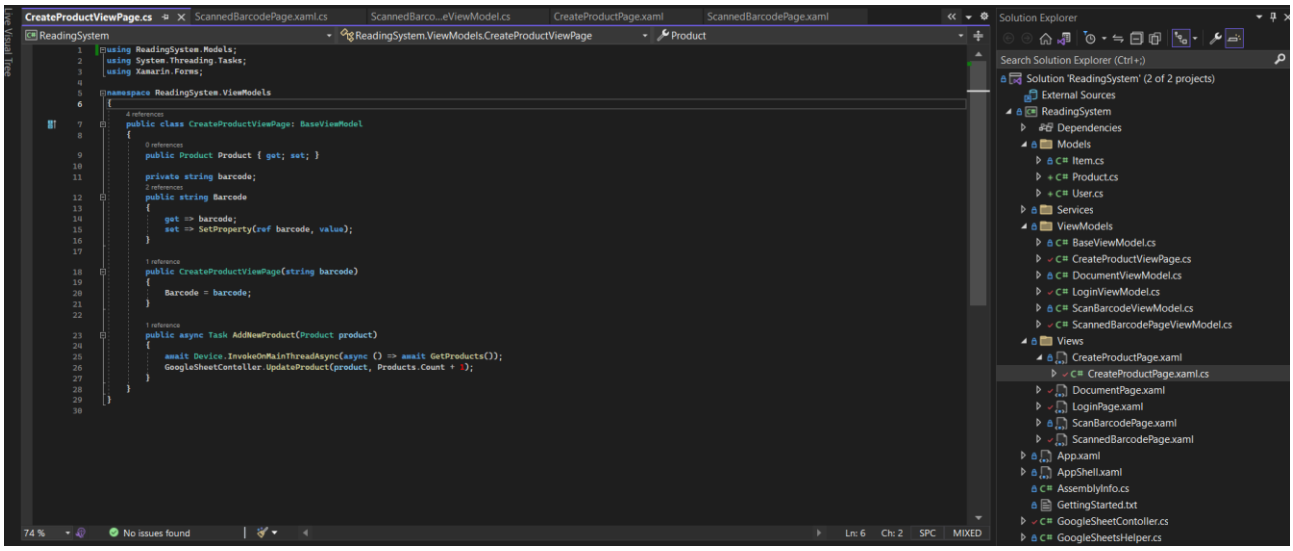

Рис.2.20 Код класу CreateProductViewPage

На рис.2.21 показано код класу CreateProductPage.xaml.cs.

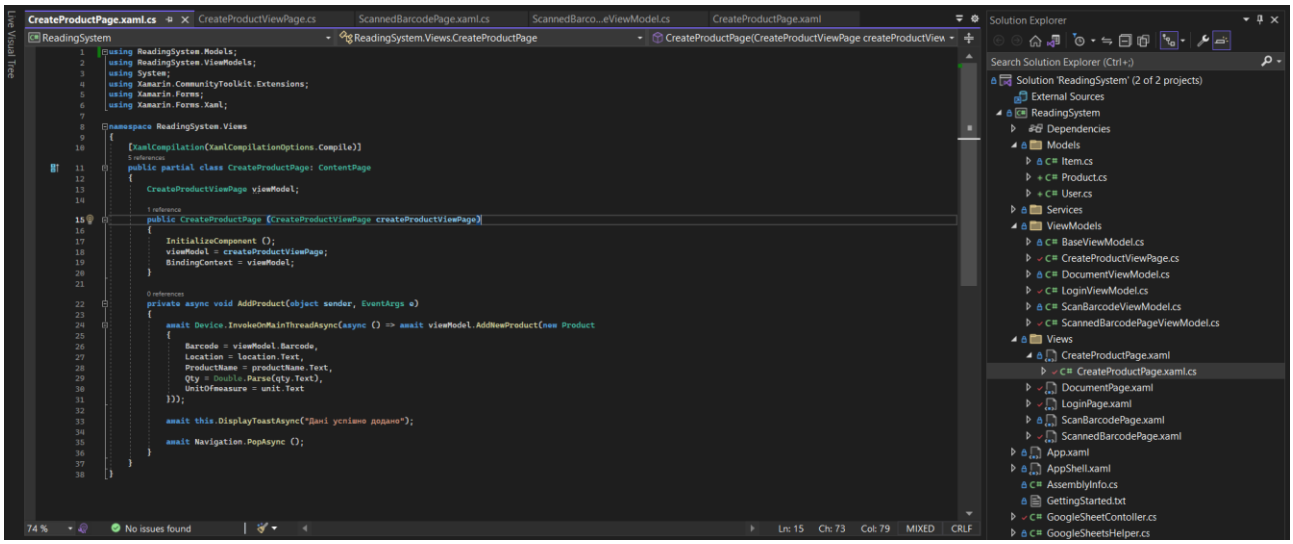

Рис. 2.21 Код файлу CreateProductPage.xaml.cs

На рис.2.22 показано вигляд сторінки створення товару.

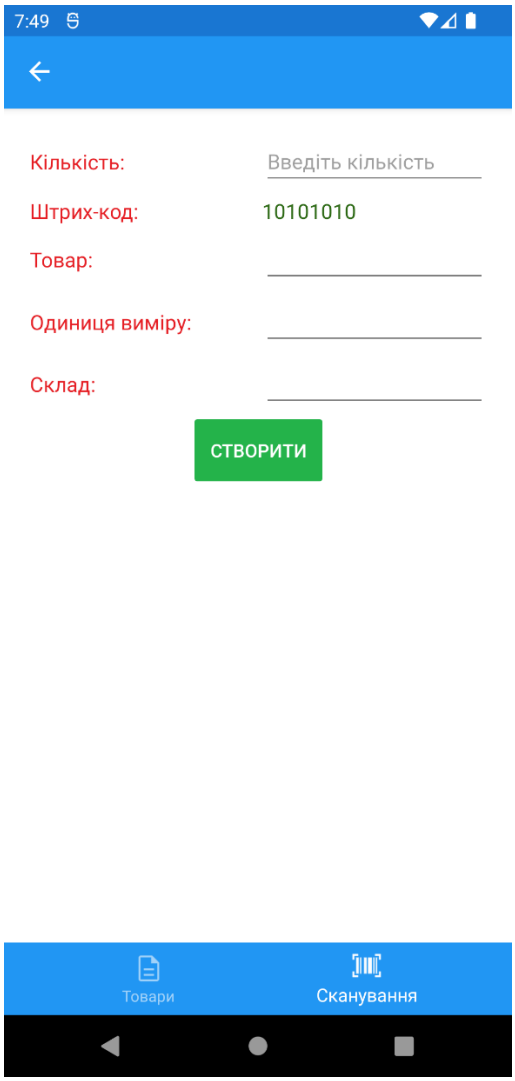

Рисунок 2.22 Екран сторінки створення нового товару

2.3 Керівництво користувача

2.3.1 Авторизація

Авторизація у додаток відбувається на спеціальній сторінки (рис.2.23), при першому заході у додаток з’являється сторінка на якій присутні такі елементи:

1. Поле введення логіну користувача.
2. Поле для введення паролю.
3. Прапорець для запам’ятовування користувача.
4. Кнопка “Увійти”.

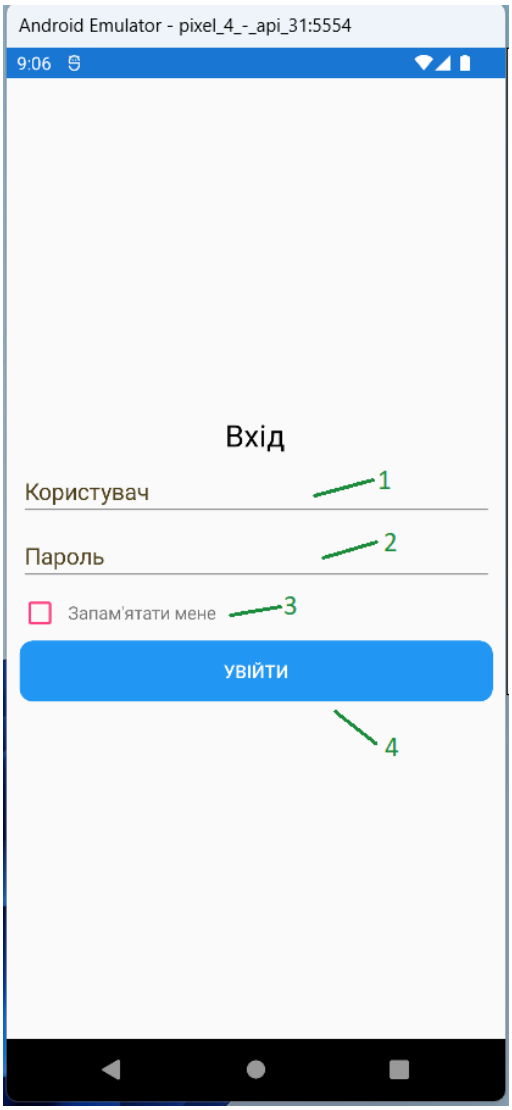

Рис. 2.23 Екран авторизації у додаток:

1- Поле введення логіну користувача; 2 – Поле введення паролю;

3 – Прапорець для запам’ятовування користувача; 4 – Кнопка “Увійти”

При натисненні на кнопку ”Увійти” відбувається авторизація користувача, перевіряється його логін (без урахування регістру) та перевіряється введений пароль(з урахуванням регістру), якщо користувач не пройде дану перевірку то у нього на екрані з’явиться повідомлення про помилку (рис. 2.24).

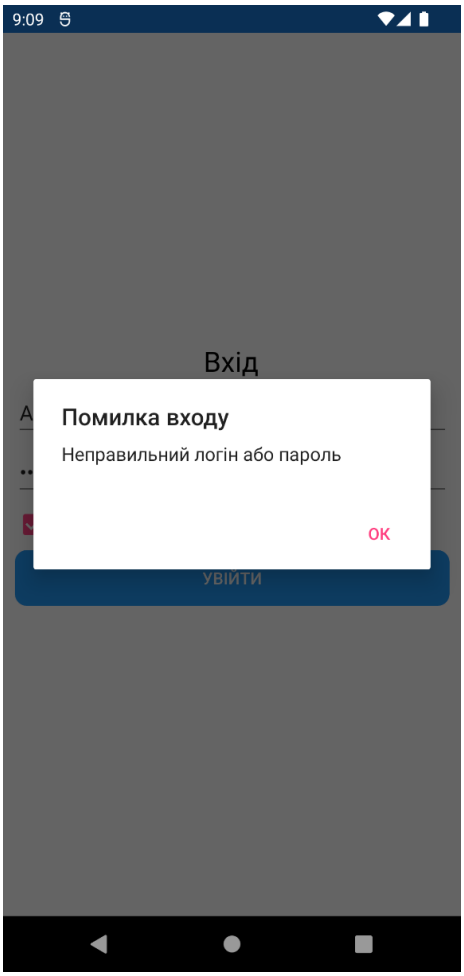

Рис. 2.24 Повідомлення про помилку при авторизації

2.3.2 Головна сторінка

При успішній авторизації у додаток вас буде направлено на головну сторінку додатку, яка містить дві сторінки а саме: сторінку товарів та сторінку сканування, перемикатись між ними можна натиснувши на відповідну сторінку на навігаційній панель (рис. 2.25, п.1) яка присутня знизу, також на цій сторінці доступна функція “Вихід” (рис. 2.25, п.2) (функція присутня тільки коли вибрано сторінку списку товарів), яка означає зміну користувача, при натисненні на неї вас перенесе на сторінку авторизації.

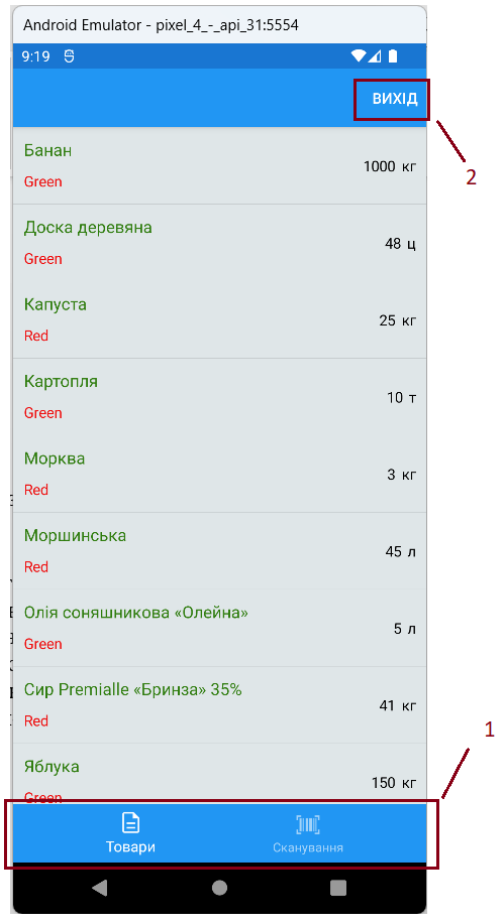

Рис. 2.25 Вигляд головної сторінки:

1 – Навігаційна панель; 2 – Кнопка “Вихід”

2.3.3 Сторінка списку товарів

Якщо у користувача на навігаційній панелі обрано “Товари” то це означає, що він перебуває на сторінці списку товарів і він буде бачити список усіх товарів (рис. 2.26, п.1).

Дана сторінка є списком “плиток” товарів і має в собі таку інформацію:

1. Назва товару - рис. 2.26, п.2.

2. Назва складу на якому зараз перебуває товар - рис. 2.26, п.3.

3. Кількість даного товару та його одиниця виміру - рис. 2.26, п.4.

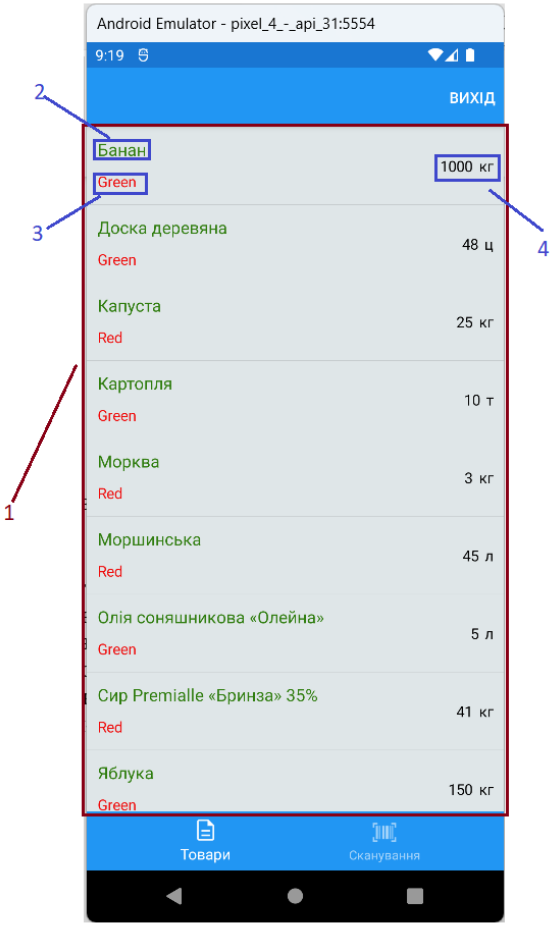

Рис. 2.26 Сторінка списку товарів: 1 – Список усіх товарів; 2 – Назва товару; 3 – Склад на якому перебуває товар; 4 – Кількість товару та його одиниця виміру

2.3.4 Сторінка редагування, додавання та видалення товару

Коли користувач на навігаційній панелі обере пункт “Сканування”, його перемістить на сторінку для сканування штрих-коду. Штрих-код можна відсканувати за допомогою камери пристрою, натиснувши на кнопку, яка знаходиться справа від поля ручного введення (рис.2.27, п.1) або ввести вручну використовуючи клавіатуру пристрою натиснувши на поле введення (рис.2.27, п.2), якщо увімкнена дана функція ручного введення штрих-коду (рис.2.27, п4). Після сканування або введення коду з’явиться кнопка “Знайти” (рис.2.27, п.3), натиснувши на неї відбудеться пошук товару за штрих-кодом.

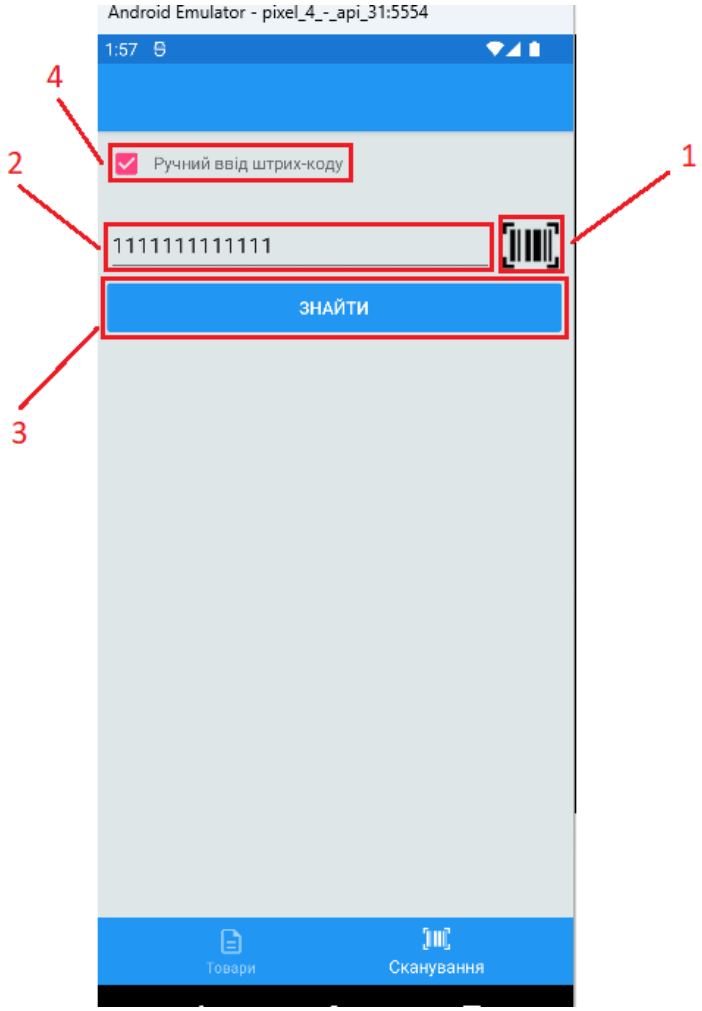

Рис. 2.27 Сторінка сканування: 1 – Кнопка для відкриття камери щоб сканувати штрих-код; 2 – Поле введення штрих-коду; 3 – Кнопка “Знайти”; 4 - Прапорець для увімкнення/вимкнення функції ручного введення штрих-коду;

Якщо механізм знайде товар по заданому штрих-коду, то на сторінці з’явиться вікно з повною інформацією про товар та можливістю змінити його кількість. Також будуть доступні 3 функції:

1. Зберегти зміни – збереже та оновить інформація за даним товаром у Google Sheets.
2. Видалити товар – видалить інформацію по даному товару з Google Sheets.
3. Очистити сторінку – очистить дану сторінку від будь яких записів, корисна для випадків коли необхідно без змін припинити обробку інформації.

Додатково: Якщо при редагуванні змінити кількість на 0, виставити прапорець поля “По товару припинено замовлення” в активний стан та зберегти зміни, то механізм видалить запис по товару!

На рис.2.28 показано екран інформації про товар.

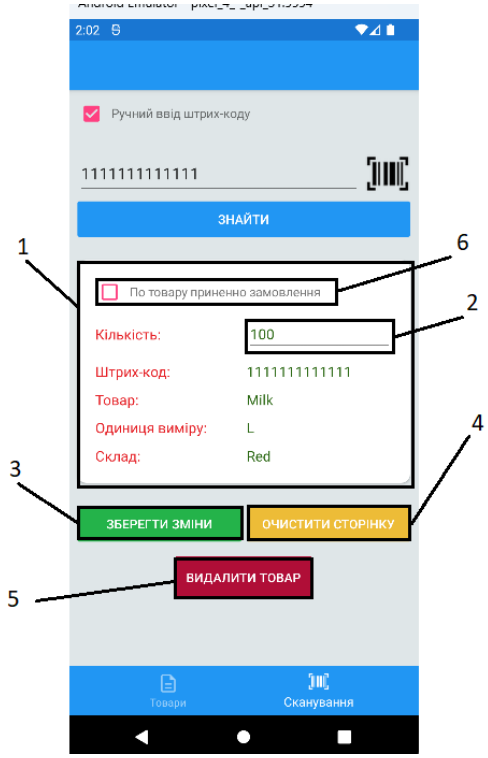

Рис. 2.28 Інформація про товар: 1 – Загальна інформація про товар; 2 – Поле для редагування кількості; 3 – Кнопка “Зберегти зміни”; 4 - Кнопка “Очистити сторінку”; 5 - Кнопка “Видалити товар”; 6 – Прапорець який регулює облік по товару;

Інформація про товар в себе включає (рис.2.28):

1. Загальну інформацію про товар (1).
2. Поле для редагування кількості (2).
3. Доступні вище вказані функції (3,4,5).
4. Параметр, який регулює облік по товару (6).

Якщо інформація за штрих-кодом не було знайдено, то з’явиться вікно з повідомленням про те що можна створити товар і прив’язати йому відсканований штрих-код. На рис.2.29 показано помилку у випадку, якщо товар не знайдено за штрих-кодом.

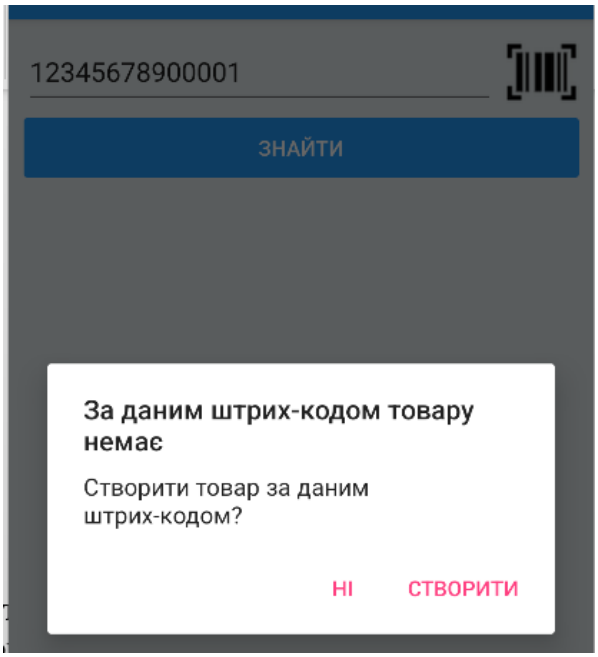

Рис. 2.29 Помилка в тому випадку, коли інформація про товар не знайдена за штрих-кодом

Якщо вибрати пункт “Створити”, то відкриється сторінка для заповнення даних товару та буде доступна кнопка “Створити”, яка збереже дану інформацію до Google Sheets таблиці.

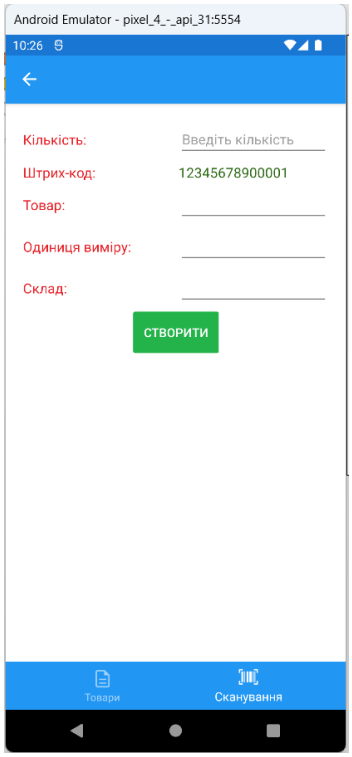

Рис. 2.30 Сторінка додавання запису

Варто звернути увагу на:

Штрих-код заповнюється автоматично і не є доступним для редагування, також цей штрих-код не генерується автоматично. Він є тим штрих-кодом, який був від сканований, але за ним не знайшовся запис.

Якщо з якихось причин перейшовши на сторінку створення запису, зникла необхідність у створені нового запису, то натиснувши на стрілку, яка знаходиться зліва угорі, то користувача верне на сторінку сканування без збереження введених значень.

ВИСНОВКИ

У ході виконання дипломної роботи було детально розглянуто поняття інформації та штрих-коду. Інформація розуміється як дані, які мають значення та можуть бути оброблені для отримання корисної інформаційної величини. Зокрема, було досліджено різні типи штрих-кодів, такі як EAN, QR-коди та Code 39, що використовуються для кодування різноманітної інформації, включаючи номери товарів, URL-адреси, текстові дані та інше.

В роботі розглядалися основні дії, які можна здійснювати з інформацією, отриманою з штрих-коду. Серед них варто виділити можливість отримання детальної інформації про товар або продукт, збереження або надсилання отриманих даних, а також взаємодію з іншими системами чи додатками для подальшого аналізу або обробки даних.

В роботі описані етапи розроблення автоматизованої системи зчитування інформації. Автоматизовану систему створено шляхом розроблення додатку до мобільного телефону на платформі Xamarin. Він здатний сканувати штрих-код і отримувати відповідний запис з Google Sheets таблиці, було показано, як інформацію, закодовану в штрих-коді, можна швидко та ефективно обробляти та використовувати. Цей підхід дає змогу автоматизувати зчитування інформації, покращує швидкість та точність обробки даних і сприяє покращенню ефективності процесів, що залежать від доступу до великого масиву даних.

Отже, розроблена автоматизована система зчитування інформації, включаючи розгляд поняття інформації та штрих-коду, типів штрих-кодів та можливих дій з отриманою інформацією, разом з розробленим мобільним додатком на платформі Xamarin, що забезпечує зручний доступ до даних з Google Sheets таблиці за допомогою сканування штрих-коду, представляє значущий внесок у сферу автоматизації та обробки інформації. Ця система може бути використана в різних галузях, де швидкий доступ та точна обробка

великого масиву даних є необхідними компонентами успішного функціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поняття інформатики [Електронний ресурс] — Режим доступу до ресурсу: https://dl.nure.ua/pluginfile.php/468/mod_resource/content/3/content/content2.html

2. Створення та архівація комп'ютерних даних - типи, призначення, друк [Електронний ресурс] — Режим доступу до ресурсу: https://www.shevchenkove.org.ua/person_syte/Golub/КТ і IT/apx.html

3. Штрих-код - типи, призначення, друк [Електронний ресурс] — Режим доступу до ресурсу: https://vizitka.com/uk/shtrih-kod

4. European Article Number [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/European_Article_Number

5. QR-код [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/QR-код

6. The global industry association that connects, standardizes and advances automatic identification technologies [Електронний ресурс] — Режим доступу до ресурсу: AIM - AIM Home (aimglobal.org)

7. Code 39 [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Code_39

8. Штрихове кодування [Електронний ресурс] — Режим доступу до ресурсу: https://www.gpp.in.ua/znaki-markuvannya/shtrikhove-koduvannya.html

9. Mobile App Development [Електронний ресурс] — Режим доступу до ресурсу: https://www.softwareag.cloud/site/product/webmethods-io-integration.html#/

10. Кольоровий штрих-код [Електронний ресурс] — Режим доступу до ресурсу: https://www.vostok.dp.ua/ukr/infa1/shtrih-kod/tsvet_shtrikh_koda/

11. Mobile Barcode Scanning and Its Use in Retail [Електронний ресурс] — Режим доступу до ресурсу: https://www.scandit.com/blog/barcode-scanning-in-mobile-apps-benefits-use-cases/

12. Microsoft Visual Studio [Електронний ресурс] — Режим доступу до ресурсу: https://visualstudio.microsoft.com/

13. What is Xamarin? [Електронний ресурс] — Режим доступу до ресурсу: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin/

14. C Sharp [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp

15. Team Foundation Server (CLR) overview [Електронний ресурс] — Режим доступу до ресурсу: https://riptutorial.com/ru/tfs/example/24947/что-такое-tfs-и-как-данные-хранятся-в-нем-/

16. Common Language Runtime (CLR) overview [Електронний ресурс] — Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/standard/clr/

17. Паттерн MVVM [Електронний ресурс] — Режим доступу до ресурсу: https://metanit.com/sharp/wpf/22.1.php

18. Google Sheets [Електронний ресурс] — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Sheets

19. Google Sheets [Електронний ресурс] — Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/Google-Spreadsheets

20. AppCenter [Електронний ресурс] — Режим доступу до ресурсу: https://devblogs.microsoft.com/appcenter/introducing-visual-studio-app-center/

21. App Center vs Azure DevOps [Електронний ресурс] — Режим доступу до ресурсу: https://smartbridge.com/app-center-vs-azure-devops/

22. Git [Електронний ресурс] — Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Git

23. What is Git? What benefits does Git offer? – Quickscrum [Електронний ресурс] — Режим доступу до ресурсу: https://guide.quickscrum.com/git-guide/